Advant® OCS
with Master software

AMPL Configuration
Advant® Controller 100 Series

Reference Manual

3BSE 009 626R401

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

ABB Master and **MasterPiece** are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland.

Advant, **AdvaBuild**, **AdvaCommand**, and **AdvaInform** are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland.

AdvaSoft, **Advant Station**, **AdvaBuild On-line Builder**, **Advant Controller**, and **Advant Fieldbus** are trademarks of ABB Asea Brown Boveri Ltd. Switzerland.

Adobe and **Acrobat** are trademarks of Adobe Systems Incorporated.

IBM and **IBM PC** are registered trademarks of International Business Machines Corporation.

Microsoft, **MS-DOS**, and **Windows** are registered trademarks, and **Windows for Workgroups** and **Windows NT** are trademarks of Microsoft Corporation.

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Automation Products GmbH. ABB Automation Products GmbH assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Automation Products GmbH be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Automation Products GmbH be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Automation Products GmbH's written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Copyright © ABB Automation Products GmbH 1999

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General Information 1-1
1.2 Purpose and Manual Organization 1-1
1.3 Definitions 1-2
1.4 Conventions 1-2
  1.4.1 General 1-2
  1.4.2 Keys 1-2
  1.4.3 Commands and Menus 1-3
1.5 Related Documentation 1-3
1.6 Release History 1-5
1.7 Terminology 1-6
1.8 Advant Controller 110 1-8
1.9 Advant Controller 160 1-8
1.10 Advant Controller 70 1-9
1.11 Advant Controller 55 1-9
1.12 Advant Station 100 Series Engineering Station 1-9

Chapter 2 - Preparations

2.1 Design of an Advant Controller 100 Series Installation 2-1
2.2 Setup of Projects and Nodes on Advant Station 100 Series ES 2-2
2.3 Directory Structure and Library Handling 2-3
2.4 Connection of Advant Station 100 Series ES to a Controller Station 2-4
  2.4.1 Examples of Connection 2-4
  2.4.2 About Server and Client Engineering Stations in Computer Network ... 2-5
  2.4.3 Overview of Remote Access Possibilities 2-5
  2.4.4 Conditions of Successful Connection 2-7
  2.4.5 Preparation of Connection 2-7

Chapter 3 - User Application Program

3.1 Overview 3-1
3.2 Generation and Storage 3-1
3.3 Application Protection 3-2
3.4 Backup and Restore 3-3
  3.4.1 Save and Backup 3-3
  3.4.2 Restore 3-3

CONTENTS (continued)

Chapter 4 - Configuration

4.1 General 4-1
- 4.1.1 Definition 4-1
- 4.1.2 Hardware of Advant Controller 100 Series 4-2

4.2 Configuration by DB Elements 4-2
4.3 Call Names of DB Elements 4-3
4.4 Graphic Symbols of DB Elements 4-5
4.5 Item Designation of DB Elements 4-5
4.6 Names of DB Elements 4-6
4.7 Configuration of DB Elements Describing the Hardware 4-6
- 4.7.1 Overview 4-6
- 4.7.2 Configuration of Unit or Module DB Elements 4-7
- 4.7.3 Channel DB Elements 4-8
- 4.7.4 Hardware Initialization 4-9

4.8 Configuration of DB Elements Describing Common Data 4-9
- 4.8.1 Overview 4-9
- 4.8.2 Configuration of DAT/PARDAT/DAT_/MDAT Elements 4-10

4.9 Communication via Advant Fieldbus 100 4-13
- 4.9.1 Overview 4-13
- 4.9.2 Configuration of DataSet Peripheral Element 4-16

4.10 Advant Fieldbus 100 Network Configurations 4-17
4.11 Communication via RCOM 4-19
- 4.11.1 Overview 4-20
- 4.11.2 Configuration of MVI Data Block Element 4-20
- 4.11.3 RCOM Network Configurations 4-21
- 4.11.4 RCOM Functions 4-23

4.12 Event Handling 4-23
- 4.12.1 Overview 4-23
  - 4.12.1.1 Event Detection 4-23
  - 4.12.1.2 Event Transfer 4-25
- 4.12.2 Configuration of Event Sets 4-33
  - 4.12.2.1 Configuration of Event Channel Elements 4-33
  - 4.12.2.2 Configuration of Event Set Elements 4-37
  - 4.12.2.3 Event Transfer via Hierarchical Network (Example) 4-42

4.13 Configuration of Extended DB Elements 4-44
- 4.13.1 Overview 4-44
- 4.13.2 Extended DB Element Communication Between Stations 4-48
- 4.13.3 Selection of Signal Sources 4-48

CONTENTS (continued)

4.13.3.1 Overview of Source Selection.. 4-49
4.13.3.2 Selecting HW (Channel Element) as Source 4-50
4.13.3.3 Selecting PC (PC Element) as Source............................ 4-51
4.13.3.4 Selecting DSP/DAT as Source 4-52
4.13.3.5 Selecting MDAT as Source for Multiple CPU Support . 4-53
4.13.3.6 Selecting HMI (MMI) as Source 4-55
4.13.3.7 Engineering Station as Source 4-55
4.13.4 Processing in Extended DB Elements 4-56
4.13.4.1 Transfer Between CALC_VAL and VALUE 4-56
4.13.4.2 Status Handling with Extended DB Elements 4-58
4.13.4.3 Event Handling with Extended DB Elements................ 4-58
4.13.4.4 Operational Mode Change Events 4-59
4.13.4.5 Error Condition Events 4-59
4.13.5 Connection to Signal Sinks.. 4-60
4.13.5.1 Connecting Extended DB Elements to HW (Channel Element) Sinks 4-60
4.13.5.2 PC Element Inputs as Extended DB Element Sinks 4-61
4.13.5.3 Selecting DSP/DAT as Sink... 4-61
4.13.5.4 Selecting MDAT as Sink for Multiple CPU Support..... 4-62
4.13.5.5 HMI (MMI) as Sink 4-63
4.13.5.6 Engineering Station as Sink 4-63
4.13.6 Miscellaneous 4-63
4.13.6.1 Connecting Extended DB Elements to PC Elements..... 4-63
4.13.6.2 Creation of DSP/DAT/EVS(S) Elements 4-64
4.13.6.3 Acknowledge Logic of HMI (MMI) Connection 4-64
4.13.7 Special Terminals of Extended DB Elements...................................... 4-65
4.13.7.1 Special Terminals in Base Part of Extended DB Elements... 4-65
4.13.7.2 Terminals in Limit Check Part of Extended DB Elements... 4-67
4.13.7.3 Special Terminals in Communication Part of Extended DB Elements... 4-68
4.13.7.4 Special Terminals HMI (MMI) Part of Extended DB Elements... 4-70
4.13.7.5 Terminals in Special Part of Extended DB Elements..... 4-70
4.13.8 Example of Extended DB Element Configuration 4-72
4.14 PROFIBUS_DP DB Elements and Communication 4-74

CONTENTS (continued)

Chapter 5 - Programming

5.1 General 5-1
- 5.1.1 PC Program 5-1
- 5.1.2 PC Elements in General 5-1
- 5.1.3 PC Diagram 5-2
- 5.1.4 Executable Unit 5-2

5.2 Data Types 5-2
- 5.2.1 Boolean 5-4
- 5.2.2 Integer 5-5
- 5.2.3 Real Numbers 5-6
- 5.2.4 Time 5-7
- 5.2.5 Group Data 5-8

5.3 Variables, Constants and Parameters 5-9
- 5.3.1 Inputs and Outputs of PC Elements 5-9
- 5.3.2 Common Variables 5-10
- 5.3.3 Constants 5-10
- 5.3.4 Function Parameters 5-10
- 5.3.5 Common Parameters 5-11
- 5.3.6 Symbolic Naming of Variables and Constants 5-11

5.4 PC Elements 5-12
- 5.4.1 Common Properties of PC Elements 5-12
  - 5.4.1.1 Representation 5-12
  - 5.4.1.2 Item Designation 5-13
  - 5.4.1.3 Call Names 5-14
  - 5.4.1.4 PC Instance Names 5-15
  - 5.4.1.5 Call Parameters 5-15
  - 5.4.1.6 Function Parameters 5-16
  - 5.4.1.7 Terminals 5-17
  - 5.4.1.8 Special Representation 5-19
- 5.4.2 Basic PC Elements 5-19
- 5.4.3 Structure PC Elements 5-19
  - 5.4.3.1 Purpose of Structuring 5-20
  - 5.4.3.2 Structuring a PC Program into PC Modules 5-20
  - 5.4.3.3 Hierarchy of the Module Headers 5-21
  - 5.4.3.4 PC Program Module PCPGM 5-21
  - 5.4.3.5 Function Modules FUNCM 5-22
  - 5.4.3.6 Control Modules CONTRM, MASTER, SEQ 5-22
  - 5.4.3.7 Master-Slave Combination MASTER/SLAVEM 5-23

CONTENTS (continued)

5.4.3.8 Sequence-Step Modules SEQ/STEP 5-23
5.4.3.9 PC Block BLOCK 5-24
5.5 PC Programming 5-25
5.5.1 Identification of PC Elements 5-25
5.5.2 Connection of Elements 5-27
5.5.3 Application with Sequences 5-32
5.5.3.1 Function Diagram 5-32
5.5.3.2 Activities 5-33
5.5.3.3 Single Sequences 5-39
5.5.3.4 Beginning of Sequence Selection (Divergence) 5-41
5.5.3.5 End of Sequence Selection (Convergence) 5-45
5.5.3.6 Complete Sequence Selections 5-47
5.5.3.7 Sequence Selection with Several Terminating Branches 5-48
5.5.3.8 Simultaneous Sequences 5-50
5.5.3.9 Sequence with Loop 5-51
5.5.3.10 Use of HOLD 5-52
5.5.3.11 Dividing a Sequence into Steps 5-54
5.5.3.12 Sequence in the Hierarchy 5-54
5.5.3.13 Function 5-56

Chapter 6 - Off-line Configuration and Programming

6.1 General 6-1
6.2 Type Circuit Handling 6-2
6.3 Translate Functions 6-3
6.3.1 Conversion from/to AMPL Source Code 6-3
6.3.2 Generation of Target Code 6-3
6.3.2.1 Restrictions 6-4
6.3.2.2 In-line Code 6-4
6.3.2.3 Target Code Generation Procedure 6-4
6.3.2.4 Memory Requirements on the Target 6-5
6.3.2.5 Example of Target Code Generation Protocol 6-6

Chapter 7 - Connecting a Controller to an Engineering Station

7.1 Connect 7-1
7.1.1 About Target Connection 7-1
7.1.2 Connect the Engineering Station to an AC 100 Series Target 7-2
7.1.3 Types of Routing, Diagrams 7-3
7.2 Target Access Protection 7-11
7.3 Disconnect 7-14

CONTENTS (continued)

Chapter 8 - Target Control Functions

8.1 General 8-1
8.2 Load Application 8-1
8.3 Block Program 8-2
8.4 Deblock Program 8-2
8.5 Restart 8-2
8.6 Save In PROM 8-2
8.7 Clear PROM 8-2
8.8 Set Date and Time 8-2
8.9 Update Slave 8-2
8.10 Upload Application 8-3
8.11 Set Password, Application Checksum 8-3
8.12 Report Status 8-4
8.13 Report Options 8-4
8.14 Report Attributes 8-5
8.15 Report Errors 8-5

Chapter 9 - On-line Configuration and Programming

9.1 About On-line and Off-line Modes and Mode Transitions 9-1
9.2 Switching to On-line Mode 9-5
9.3 Modifying the Application Program On-line 9-6
9.4 On-line Editing Rules 9-6

Chapter 10 - Test Functions

10.1 General 10-1
10.2 Configuration Part 10-1
  10.2.1 Value Display (Monitoring) of DB Terminals 10-2
  10.2.2 Forcing DB Terminals 10-2
  10.2.3 Setting DB Element Terminals 10-3
10.3 Program Part 10-3
  10.3.1 Value Display (Monitoring) 10-4
  10.3.2 Forcing PC Terminals 10-4

Chapter 11 - Execution

11.1 Internal Software of Advant Controller 100 Series 11-1
  11.1.1 Kernel 11-2
  11.1.2 Communication 11-2
  11.1.3 Diagnostic Program 11-2

CONTENTS (continued)

  - 11.1.4 Function Block Library 11-2
- 11.2 Process I/O Handling 11-3
- 11.3 Input and Output Phases 11-3
  - 11.3.1 Input 11-3
  - 11.3.2 Output 11-4
- 11.4 Execution Order 11-4
- 11.5 Execution of Elements 11-5
- 11.6 Reset Execution 11-6
- 11.7 Initialization 11-7
- 11.8 On-line Changes 11-7
  - 11.8.1 Configuration Part 11-7
  - 11.8.2 Program Part 11-8
- 11.9 Test 11-9
  - 11.9.1 Configuration Part 11-9
  - 11.9.2 Program Part 11-10

Appendix A - Application Examples

- A.1 Control of Lift A-1
- A.2 Control of a Car Position A-7

Appendix B - Data Flow in Advant Station 100 Engineering Station

- B.1 General B-1
- B.2 Files B-1
- B.3 Data Flow Diagram B-2

Appendix C - Glossary

INDEX

ILLUSTRATIONS

- Figure 1-1. Example AC 110, Basic Station with Optional Extension Backplane 1-8
- Figure 1-2. Example AC 160, Basic Station with Redundancy and Optional Extension Backplane 1-8
- Figure 1-3. Advant Controller 70 Station 1-9
- Figure 1-4. AC 55 Basic Unit (CT481) with 2 I/O Expansion Units (CX473/CX475) 1-9
- Figure 2-1. Design of an Advant Controller 100 Series Installation (Flow Diagram) 2-1
- Figure 2-2. Project/Node Structure on Advant Station 100 Series ES, Example 2-2

ILLUSTRATIONS (continued)

Figure 2-3. Project and Node Definition on AS 100 Series ES, Example 2-2
Figure 2-4. Direct Connection of AS 100ES to Advant Controller 100 Series.............. 2-4
Figure 2-5. Routing from AS 100ES to AC 110 and 70 via AF 100, Example 2-4
Figure 2-6. Routing from AS 100ES to AC 110 via RCOM, Example.......................... 2-4
Figure 3-1. User Application Program for Advant Controller 100 Series...................... 3-1
Figure 4-1. Configuration Part Within User Application Program 4-1
Figure 4-2. Engineering Station View of Basic Unit (CT481, Example)........................ 4-7
Figure 4-3. Engineering Station View of Channel, Base Part (AI620, Example).......... 4-8
Figure 4-4. Engineering Station View of DAT(R).. 4-11
Figure 4-5. DAT_DAT Usage to Convey Names to HMI 4-13
Figure 4-6. DataSet Communication, Sending Station... 4-14
Figure 4-7. DataSet Communication, Receiving Station 4-15
Figure 4-8. Representation of DSP Element, Overview.. 4-16
Figure 4-9. AC 110 and 70 Stations Connected via Advant Fieldbus 100.................... 4-18
Figure 4-10. DataSet Communication via Advant Fieldbus 100 (Example).................. 4-19
Figure 4-11. Representation of MVB Element, Overview... 4-20
Figure 4-12. RCOM Network Configuration (Example) 4-22
Figure 4-13. Hierarchical RCOM Network Configuration (Example)........................... 4-22
Figure 4-14. Event Detection for PC Signals of Type Real.. 4-24
Figure 4-15. Configuration Alternatives for Event Transfer 4-26
Figure 4-16. Event Set (EVS) Element Relationship 4-28
Figure 4-17. Event Channel Relationship Advant Controller 400 Series and AC 110 or 160... 4-29
Figure 4-18. Event Channel Relationship AC 400 Series and AC 70 or AC 55............ 4-29
Figure 4-19. Event Transfer via AF 100 from AC 110 and 70 to AC 400 Series (Example)... 4-30
Figure 4-20. Event Transfer via RCOM from AC 110 and 55 to AC 400 Series (Example)... 4-31
Figure 4-21. Event Transfer from AC 110, 70, and 55 to PC with AdvaSoft for Windows (Example)... 4-32
Figure 4-22. AIC Element, Base Part (Engineering Station View)................................ 4-33
Figure 4-23. AIC Element, Limit Check (Engineering Station View) 4-34
Figure 4-24. DIC Element (Engineering Station View) 4-34
Figure 4-25. DI650/DI651/DI652 Module, Base Part (Engineering Station View)........ 4-35
Figure 4-26. DI650/DI651/DI652 Module, Parameter Part (Engineering Station View)4-36
Figure 4-27. DI650/DI651/DI652 Module, SOE Handler (Engineering Station View).. 4-36
Figure 4-28. DIS650/DIS651/DIS652 Channel Element (Engineering Station View) ... 4-37
Figure 4-29. EVS(S) Element, Base Part (Engineering Station View) 4-38
Figure 4-30. EVS(S), Event Channel References 1-16 (Example) 4-38

ILLUSTRATIONS (continued)

Figure 4-31. Relation Between EVS(S) and Event Channel Elements (Example)......... 4-39
Figure 4-32. EVS(T) Element (Engineering Station View)... 4-40
Figure 4-33. Extended DB Element, System Components with Advant OPC Server 4-46
Figure 4-34. Extended DB Element, System Components with AdvaSoft for Windows4-47
Figure 4-35. Extended DB Element Referencing HW Element..................................... 4-48
Figure 4-36. HW as Source of Extended DB Element... 4-50
Figure 4-37. PC as Source of Extended DB Element 4-51
Figure 4-38. DSP/DAT as Source of Extended DB Element... 4-53
Figure 4-39. MDAT as Source of Extended DB Element.. 4-54
Figure 4-40. HMI (MMI) as Source of Extended DB Element 4-55
Figure 4-41. Extended DB Element Data Flow 4-57
Figure 4-42. HW as Sink of Extended DB Element 4-60
Figure 4-43. DSP/DAT as Sink of Extended DB Element... 4-62
Figure 4-44. MDAT as Sink of Extended DB Element.. 4-63
Figure 4-45. Example of Extended DB Element Configuration 4-73
Figure 5-1. Program Part Within User Application Program.. 5-1
Figure 5-2. Example of a PC Element in Graphical Representation............................... 5-2
Figure 5-3. Delay Element TON 5-3
Figure 5-4. Counter Element COUNT 5-3
Figure 5-5. PC Element AND with Three Inputs.. 5-4
Figure 5-6. Multiplexer MUX-MI (B,3) 5-4
Figure 5-7. PC Element COUNT 5-5
Figure 5-8. PC Element COMP-I.. 5-5
Figure 5-9. PC Element PI 5-6
Figure 5-10. TON PC Element.. 5-7
Figure 5-11. PC Element INT 5-7
Figure 5-12. PC Element FUNG-1V... 5-8
Figure 5-13. PC Element REG-G.. 5-9
Figure 5-14. Constant at PC Element TON.. 5-10
Figure 5-15. Example of Boolean Constants 5-10
Figure 5-16. Symbolic Name of Output Variable 5-11
Figure 5-17. Symbolic Name of a Constant...5-11
Figure 5-18. PC Element Symbol 5-12
Figure 5-19. Example of Element with Function Symbol 5-12
Figure 5-20. Example of Element with Standard Symbol 5-12
Figure 5-21. Example of the Item Designation System 5-13
Figure 5-22. Function Parameter of PC Element PI... 5-16
Figure 5-23. Example of Terminal Numbers of an AND Element 5-17

ILLUSTRATIONS (continued)

Figure 5-24. Terminal Names of PC Element COUNT.. 5-17
Figure 5-25. Symbol for Dynamic Input 5-18
Figure 5-26. Symbol for Inverted Input Signal 5-19
Figure 5-27. Module Header in a PC Module 5-20
Figure 5-28. The Two Cases for Composing PC Modules... 5-21
Figure 5-29. Example of a Tree Form 5-25
Figure 5-30. Example of a Division into Modules 5-26
Figure 5-31. Inverted Signal... 5-27
Figure 5-32. Output Variables of PC Element COUNT 5-29
Figure 5-33. PC Element Connection Within the Same Page and Execution Unit 5-30
Figure 5-34. PC Element Connection at Different Pages but the Same Execution Unit. 5-30
Figure 5-35. PC Element Connection at Different Pages and Different Execution Units5-30
Figure 5-36. Connection to DB Element in a PC Diagram 5-31
Figure 5-37. Examples of Transition Conditions 5-32
Figure 5-38. Activities.. 5-33
Figure 5-39. Examples of N Order 5-33
Figure 5-40. Examples of Continuous Order with Condition 5-34
Figure 5-41. Example of Delayed Activity 5-34
Figure 5-42. Step Module Header for Delayed Activity, Example 5-35
Figure 5-43. Examples of Time-Limited Activity.. 5-35
Figure 5-44. Example of Step Being Active for a Certain Time.................................... 5-36
Figure 5-45. Example of Activity with Memory Function... 5-37
Figure 5-46. Additional Example of Activity with Memory Function........................... 5-38
Figure 5-47. Example of Single Sequence 5-39
Figure 5-48. Automatic Resetting 5-40
Figure 5-49. Automatic Resetting with Restart.. 5-40
Figure 5-50. Automatic Resetting Without Restart 5-40
Figure 5-51. Sequence Selection 5-41
Figure 5-52. Branch in Advant Controller.. 5-42
Figure 5-53. Branch.. 5-42
Figure 5-54. Implementation of the Branch in Figure 5-53 5-43
Figure 5-55. Branching Controlled by a Selector... 5-44
Figure 5-56. End of Sequence Selection 5-45
Figure 5-57. Implementation of the Convergence in Figure 5-56a 5-46
Figure 5-58. Combined Branch and Convergence and Their Implementation............... 5-47
Figure 5-59. Branch with Terminal Branches 5-48
Figure 5-60. Implementation of a Sequence According to Figure 5-59......................... 5-49
Figure 5-61. Simultaneous Sequences and Their Implementation................................. 5-50

ILLUSTRATIONS (continued)

Figure 5-62. Sequence Loop and Its Implementation 5-51
Figure 5-63. Example of Manual Control 5-53
Figure 5-64. Hierarchical Subdivision 5-54
Figure 5-65. Sequence and Step in the Hierarchy 5-55
Figure 5-66. Function Diagram for Step Change 5-56
Figure 5-67. Step Change 5-58
Figure 5-68. Example of Use of the RUN Signal 5-59
Figure 5-69. Output Whose Value Remains when the Step is Left 5-59
Figure 5-70. Connection of the RUN Output of a Step to the Data Base 5-60
Figure 6-1. Implementation and Testing of a Program for AC 100 Series 6-2
Figure 6-2. Target Code Generation Protocol (Example) 6-6
Figure 7-1. Point-to-point Connection via COMx, Topology 7-4
Figure 7-2. Point-to-point Connection via COMx, Setting 7-5
Figure 7-3. Routing via COMx, AC 100 Series Node, AF 100, Topology 7-5
Figure 7-4. Routing via COMx, AC 100 Series Node, AF 100, Setting 7-5
Figure 7-5. Routing via COMx, AC 100 Series, AF 100, 2nd AC 100 Series, 2nd AF 100, Topology 7-6
Figure 7-6. Routing via COMx, AC 100 Series, AF 100, 2nd AC 100 Series, 2nd AF 100, Setting 7-6
Figure 7-7. Routing via X3, AC 4xx, AF 100, Topology 7-6
Figure 7-8. Routing via X3, AC 4xx, AF 100, Setting 7-7
Figure 7-9. Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100, Topology 7-7
Figure 7-10. Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100, Setting 7-7
Figure 7-11. Routing via X3, AC 4xx, RCOM Net, Topology 7-8
Figure 7-12. Routing via X3, AC 4xx, RCOM Net, Setting 7-8
Figure 7-13. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, Topology 7-8
Figure 7-14. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, Setting . 7-9
Figure 7-15. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100, Topology 7-9
Figure 7-16. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100, Setting 7-9
Figure 7-17. Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net, Topology 7-10
Figure 7-18. Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net, Setting 7-10
Figure 7-19. Routing via AF 100, Topology 7-10
Figure 7-20. Routing via AF 100, Setting 7-11

ILLUSTRATIONS (continued)

Figure 7-21. Set Initial Password on Target 7-12
Figure 7-22. Target Password Query at Target Connection 7-12
Figure 7-23. Change Password on Target 7-13
Figure 7-24. Clear Password on Target 7-13
Figure 8-1. Status Report Window, Example 8-4
Figure 8-2. Report Options Window (Example, AC 55) 8-5
Figure 8-3. Report Attributes Dialog Box (Example, AC 110) 8-5
Figure 8-4. Report Error Window (Example, AC 110 2.0) 8-6
Figure 9-1. FCB Modes (States) and Transitions 9-2
Figure 11-1. Block Diagram of AC 100 Series100 Series Internal Software 11-1
Figure 11-2. Example for the Necessity of Having Input Elements 11-4
Figure 11-3. Execution of an ADD Element 11-6
Figure 11-4. Comparator COMP-I 11-6
Figure A-1. Control of Lift A-1
Figure A-2. Block Diagram of Process A-2
Figure A-3. Logic Diagram of Lift Control A-3
Figure A-4. PC Diagram with Two PC Modules, Draft A-4
Figure A-5. Completed PC Diagram of Lift Control A-6
Figure A-6. Car Control A-7
Figure A-7. Flow Diagram for Car Sequence Control A-8
Figure A-8. Function Diagram for Car Control A-9
Figure A-9. Car Sequence Implemented in an AC 100 Series System A-10
Figure B-1. Data Flow on the Engineering Station B-2

TABLES

Table 1-1. Common Documents 1-3
Table 1-2. Controller Related Documents 1-4
Table 2-1. Routing Possibilities Depending on Node Type 2-6
Table 4-1. Examples of Call Names Advant Controller 110 4-3
Table 4-2. Examples of Call Names Advant Controller 70 4-4
Table 4-3. Examples of Call Names Advant Controller 55 4-4
Table 4-4. Examples of Item Designation 4-6
Table 4-5. DB Elements Representing Common Data Area 4-10
Table 4-6. Bits in EVS Terminal ERRTYPE 4-41
Table 4-7. Terminals of EVS(S) Element EVS1 4-42

TABLES (continued)

Table 4-8. Terminals of EVS(S) Element EVS1 4-42
Table 4-9. Terminals of EVS(T) Element EVS2 4-42
Table 4-10. Terminals of EVS(S) Element EVS1 4-43
Table 4-11. Terminals of EVS(R) Element EVS1 4-43
Table 4-12. Terminals of EVS(R) Element EVS2 4-43
Table 4-13. Terminals of EVS(R) element EVS3... 4-44
Table 4-14. Function and Features of Extended DB Elements...................................... 4-49
Table 4-15. Operational Mode Change Events.. 4-59
Table 4-16. Error Condition Events... 4-59
Table 4-17. ACT Terminal... 4-65
Table 4-18. SOURCE Terminal... 4-65
Table 4-19. HW_REF Terminal... 4-65
Table 4-20. VALUE Terminal.. 4-66
Table 4-21. CALC_VAL Terminal 4-66
Table 4-22. STATUS Terminal 4-67
Table 4-23. CALC_ERR Terminal 4-67
Table 4-24. HI_LIM, LO_LIM Terminals.. 4-67
Table 4-25. DS_COM Terminal 4-68
Table 4-26. DSP_STAT Terminal... 4-68
Table 4-27. DSP_IDNT Terminal... 4-68
Table 4-28. DSP_REF Terminal... 4-68
Table 4-29. EV_COM Terminal 4-69
Table 4-30. EVS_IDNT Terminal.. 4-69
Table 4-31. EVS_REF Terminal.. 4-69
Table 4-32. MDAT_COM Terminal 4-69
Table 4-33. MDAT_ID Terminal 4-70
Table 4-34. MMI_USE Terminal... 4-70
Table 4-35. HW_PCDIA Terminal.. 4-70
Table 4-36. LIM_SRC Terminal.. 4-71
Table 4-37. SCALING Terminal 4-72
Table 5-1. Designations in Data Sheet.. 5-18
Table 6-1. Fixed Restrictions... 6-4
Table 7-1. Position of Intermediate Nodes 7-4
Table 7-2. Position of Target.. 7-4
Table 9-1. FCB Mode (State) Transitions... 9-4

Chapter 1 Introduction

1.1 General Information

This reference manual applies to configuration and application programming of Advant Controller 100 Series and their derivatives.

This version of the manual describes also the concept and usage of extended DB elements.

Function Chart Builder running on Advant Station 100 Series Engineering Stations serves as tool and platform for configuration and application programming for all these controllers.
In case of Advant Controller 110 and 70 and off-line work also Advant Station 500 Series Engineering Stations can serve as platform to run AdvaBuild Function Chart Builder on.

This manual also covers commissioning and testing.

1.2 Purpose and Manual Organization

This manual provides you with off-line and on-line methods and **common** aspects of configuration and application programming of Advant Controller 100 Series using Function Chart Builder running on Advant Station 100 Series Engineering Stations.
Function Chart Builder (a 32-bit application) is a component of AMPL Control Configuration under Windows NT.

The described off-line methods and common aspects of configuration and application programming of Advant Controller 110 and 70 apply also work using AdvaBuild Function Chart Builder running on Advant Station 500 Series Engineering Stations.
Methods and aspects specific to the controllers find in the user’s guides of Advant Controller 100 Series. See Section 1.5, Related Documentation.

Chapters

Chapter 2, Preparations and Chapter 3, User Application Program give a brief overview of topics covered in the manual.

Chapter 4, Configuration describes configuration.

Chapter 5, Programming describes application programming in general.

Chapters 7 to 10 describe configuration, application programming, and testing in practice.

Appendices

Appendix A provides small application examples.

Appendix B shows the Data Flow in Advant Station 100 Series Engineering Station.

Appendix C, the Glossary, contains definitions of terms.

1.3 Definitions

In an engineering station, a complete Advant Controller 100 Series station as part of a project is represented by a *node*.

In case of communication between engineering station and Advant Controller 100 Series station, the latter are referred as a *target*.

The user-written part of the software for a node is called the *user application program*. A user application program consists of the following parts:

- *Configuration part* (DB elements of the related node)
- *Program part* (PC elements of the related PC program).

The term *configuration* within this manual refers to the creation and modification of DB elements (configuration from a software point of view).

The term *programming* within this manual refers to the creation and modification of PC programs, including connections within the PC program and to DB elements.

1.4 Conventions

1.4.1 General

Books, manuals, and so on, referenced in the text are written in *italics*.
Terms defined in Glossary or in Section 1.7, Terminology are in *italics*.

Filenames, directory names, acronyms appear in all CAPITALS.

Words, which you have to type exactly in that form, are shown in **boldface** type, and if possible, in CAPITAL letters, for example, **POWERSTN**.

Texts and system messages displayed on the screen are shown in `Courier` font, for example, `Select Options`.

Application windows, document windows, and dialog boxes are referenced by their names in **boldface** type, for example, **PC Section** window.

1.4.2 Keys

Special keyboard keys are written plain with small capitals:

- ALT, CTRL, DEL, ENTER, ESC, INS, KEYPAD 5, NUMLOCK, SCROLL LOCK, SHIFT, SPACE
- BACKSPACE (for <-)
- DOWN (for down arrow, numeric keypad 2), LEFT (for left arrow, numeric keypad 4), UP (for up arrow, numeric keypad 8), RIGHT (for right arrow, numeric keypad 6)
- F1**,** F2, and so on (for function keys)
- MINUS (for ‘-’ sign in numeric keypad), PLUS (for ‘+’ sign in numeric keypad), ASTERISK (for ‘*’ sign in numeric keypad)
- PGDOWN (for page down, numeric keypad 9), PGUP (for page down, numeric keypad 3)
- 1, 2,... (for keys in the upper row of the traditional typewriter keyboard, not in the numeric keypad)

Shorthand for combined key usage, such as holding down the CTRL key and then pressing the key L:

CTRL+L

1.4.3 Commands and Menus

Commands are written bold with capital initial:

choose **Exit** …

Menus are written bold with capital initial:

in the **File** menu …

Shorthand for commands that you choose as items from menus, for example, command **Exit** from the **File** menu, appears as **File | Exit**.

1.5 Related Documentation

Common documents are listed in Table 1-1.

Table 1-1. Common Documents

Manual	Description
Advant Station 100 Series Engineering Station User's Guide	Describes the hardware and setup of Advant Station 100 Series Engineering Stations, including the Advant Station 100 Series Engineering Board.
Advant Station 500 Series Engineering Station User's Guide	Contains technical descriptions of the AS 500ES specific functions.
Application Builder User's Guide	Describes installation of Advant Engineering components and contains the operating instruction of Application Builder and Advant Engineering support functions.
Function Chart Builder User's Guide	This guide describes how Function Chart Builder can be used to configure different target systems.
Bus Configuration Builder User's Guide	Describes bus configuration support functions for AF100 OPC based and AdvaSoft for Windows based Human-Machine Interface systems.
Source Code Handling User's Guide	Contains descriptions and instructions for use of AMPL source code for application programs and data bases.
Advant Fieldbus 100 User's Guide	This guide describes how to install and use the Advant Fieldbus 100 network. It also provides a technical description of available communication boards and modems.
S800 I/O User's Guide	Contains user information on S800 I/O.
PROFIBUS Library Editor User's Guide	Describes the usage of the PROFIBUS Library Editor.

Controller related documents are listed in Table 1-2.

Table 1-2. Controller Related Documents

Manual	Description
Advant Controller 110 User's Guide	Contains a description of the Advant Controller 110.
Data Base Elements AC 100 Series Reference Manual	Data for all data base elements in Advant Controller 110.
PC Elements AC 100 Series Reference Manual	Data for all PC elements in Advant Controller 110.
RCOM Advant Controller 110 User's Guide	Contains technical descriptions, instructions for configuration, installation, and fault-tracing of RCOM in AC 110.
Advant Controller 160 User's Guide	Contains a description of the Advant Controller 160.
Data Base Elements Advant Controller 160 Reference Manual	Data for all data base elements in Advant Controller 160.
PC Elements Advant Controller 160 Reference Manual	Data for all PC elements in Advant Controller 160.
Advant Controller 70 User's Guide	Contains a description of the Advant Controller 70.
Data Base Elements Advant Controller 70 Reference Manual	Data for all data base elements in Advant Controller 70.
PC Elements Advant Controller 70 Reference Manual	Data for all PC elements in Advant Controller 70.
Advant Controller 55 User's Guide	Contains a description of the Advant Controller 55.
Data Base Elements Advant Controller 55 Reference Manual	Data for all data base elements in Advant Controller 55.
PC Elements Advant Controller 55 Reference Manual	Data for all PC elements in Advant Controller 55.

Table 1-2. Controller Related Documents (Continued)

Manual	Description
RCOM Advant Controller 55 User's Guide	Contains technical descriptions, instructions for configuration, installation, and fault-tracing of RCOM in AC 55.
Advant Controller 110 PROFIBUS-DP User's Guide	Describes the usage of PROFIBUS-DP (DP/E) interface implemented in the Advant Controller 110.

1.6 Release History

This is version 4 of the AMPL configuration document **common** for Advant Controller 100 Series. It takes new features of programming tools into account.

This document and the user's guides of Advant Controller 100 Series supersede the predecessor document and the earlier controller-specific documents.

It applies to the following controllers:

- Advant Controller 55 versions 1.0 and 1.1
- Advant Controller 70 versions 1.0 to 1.2
- Advant Controller 110 versions 2.0 to 2.3.
- Advant Controller 160 versions 1.0 to 2.0.
- Other Advant Controller 100 Series derivatives, such as Advant Controller 166 and Advant Controller 80, concerning the common concepts, methods, and properties.

Predecessor documents:

1. 3BSE 009 626R0001

   First common document for Advant Controller 110, 70, and 55.

2. 3BSE 009 626R0101

   Common document for Advant Controller 110, 70, and 55. Introduces extended DB elements.

3. 3BSE 009 626R0201

   Improved common document for Advant Controller 110, 70, and 55.

4. 3BSE 009 626R0301

   Improved common document for Advant Controller 110, 70, and 55.

Earlier predecessor documents (different documents for different Advant Controllers):

1. 3BSE 006 527R0001

   Initial version for MasterPiece 55 version 1.0 and Function Chart Editor (predecessor of AdvaBuild for Windows Function Chart Builder).

2. 3BSE 006 527R0101

   Version for Advant Controller 55 version 1.0 and adapted to AdvaBuild for Windows Function Chart Builder.

3. 3BSE 000 503R0001

   Initial version for MasterPiece 90 version1.0 (predecessor of Advant Controller 110) and Function Chart Editor (predecessor of AdvaBuild for Windows Function Chart Builder).

4. 3BSE 000 503R0101

   Initial version for Advant Controller 110 version 2.0 and Function Chart Editor.

5. 3BSE 000 503R0201

   Version for Advant Controller 110 version 2.0 adapted to AdvaBuild for Windows Function Chart Builder.

1.7 Terminology

The following is a list of terms you should be familiar with.

Term	Description
AC 110	The abbreviation “AC 110” is used for Advant Controller 110.
AC 160	The abbreviation “AC 160” is used for Advant Controller 160.
AC 55	The abbreviation “AC 55” is used for Advant Controller 55.
AC 70	The abbreviation “AC 70” is used for Advant Controller 70.
AC 80	The abbreviation “AC 80” is used for Advant Controller 80.
Advant Controller 100 Series/AC 100 Series	The term “Advant Controller 100 Series” is used for Advant Controller *X* where *X* may have a value from 1 to 199.
Advant Controller 400 Series/AC 400 Series	The terms “Advant Controller 400 Series” and “AC 400 Series” are used for Advant Controller 410 and Advant Controller 450.
Advant Station 100 Series Engineering Stations	The term “Advant Station 100 Series Engineering Stations” and “AS 100ES” are used for Advant Station 120 Engineering Station (AS 120ES) and Advant Station 130 Engineering Station (AS 130ES).
Advant Station 500 Series Engineering Stations	The term “Advant Station 500 Series Engineering Stations” and “AS 500ES” are used for Advant Station 515 Engineering Station (AS 515ES) and Advant Station 520 Engineering Station (AS 520ES).
AF 100	The abbreviation “AF 100” is used for Advant Fieldbus 100.

Term	**Description**
AF100 OPC	The abbreviation “AF100 OPC” is used for AF 100 OLE Process Control, an interface between AF 100 bus and HMI system.
APB	The abbreviation “APB” is used for Application Builder.
Application Builder	The term is used for Application Builder, an Advant Engineering product component.
BCB	The abbreviation “BCB” is used for Bus Configuration Builder.
Bus Configuration Builder	The term is used for Bus Configuration Builder, an Advant Engineering product component serving for communication for Human-Machine Interface systems.
Controller	The term controller is used for Advant Controller 100 Series and their derivatives.
DB	The term “DB” is used for Advant Controller 100 Series data base. It consists of database elements (DB elements).
Engineering board	The term *engineering board* is used for the Advant Station 100 Series Engineering Board - ISA and Advant Station 100 Series Engineering Board - PCI.
Engineering station	The term *engineering station* is basically used for the Advant Station 100 Series Engineering Station. In case of Advant Controller 110 and 70 and off-line work, *engineering station* can be also used for Advant Station 500 Series Engineering Stations.
FCB	The abbreviation “FCB” is used for Function Chart Builder.
Function Chart Builder	The term is used for the Function Chart Builder, an Advant Engineering product component. It is running on Advant Station 100 Series Engineering Stations.
HMI	The abbreviation “HMI”—a substitute for MMI (Man-Machine Interface—is used for Human-Machine Interface system, such as AdvaSoft for Windows or AF100 OPC.
Series	The term “Series” in product names is used to reference the products in general terms, without any reference to a specific model (for example, Advant Station 100 Series Engineering Stations).

1.8 Advant Controller 110

The Advant Controller 110 is a modular programmable logical control system for process control.

PM632	CI615	DI610	DI610	DI610	DI610	DI610	DI610	DI610	SA610
DI610	DI610	DI610	DI610	DI610	DI610	DI610	AI610	AI610	AO610

Figure 1-1. Example AC 110, Basic Station with Optional Extension Backplane

1.9 Advant Controller 160

The Advant Controller 160 is a modular programmable logical control system for process control.

CI630	CI630	PM645	PM645	PM645	DI610	DI610	DI610	DI610	SA610
DI610	DI610	DI610	DI610	DI610	DI610	DI610	AI610	AI610	AO610

Figure 1-2. Example AC 160, Basic Station with Redundancy and Optional Extension Backplane

1.10 Advant Controller 70

Advant Controller 70 is a small, powerful process station for logical control.

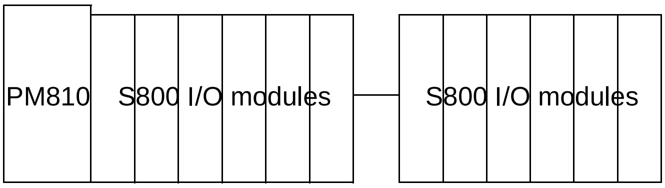


Figure 1-3. Advant Controller 70 Station

1.11 Advant Controller 55

Advant Controller 55 is a compact remote terminal unit and programmable logical control system for process control.

Figure 1-4. AC 55 Basic Unit (CT481) with 2 I/O Expansion Units (CX473/CX475)

1.12 Advant Station 100 Series Engineering Station

Advant Station 100 Series Engineering Station (AS 100ES) with the Advant Engineering components Application Builder and Function Chart Builder is a high performance tool for configuration, programming, documentation, testing, and commissioning of—among others—Advant Controller 100 Series.

AS 100ES is based on an IBM PC-AT compatible personal computer hardened for industrial purposes. The communication of AS 100ES with the Advant Controller 100 Series works via a standard serial V.24 interface with a baud rate of 9600 baud.

NOTE

Advant Station 100ES, directly connected to some Advant Controller, can be remotely connected to other Advant Controller 100 Series process stations via MasterBus 300, Advant Fieldbus 100, and RCOM network. For connection routing, see Section 2.4, Connection of Advant Station 100 Series ES to a Controller Station.

Advant Engineering offers several editors and other tools for configuration and programming. The user interface of the Advant Engineering components correspond to Windows conventions.

NOTE

The On-line Builder (earlier command-oriented functions) do not apply to Advant Controller 100 Series.

Chapter 2 Preparations

2.1 Design of an Advant Controller 100 Series Installation

The flow diagram in Figure 2-1 shows you the procedure to follow to design an Advant Controller 100 Series installation.
The scope of the control equipment is specified first.
The arrangement of the I/O channels is then determined and the plant circuit diagram can be drawn. The work can then be divided into two parts:

- Hardware part
  Ordering necessary equipment, and so on
- Software part
  Development of the user application program, called configuration and programming

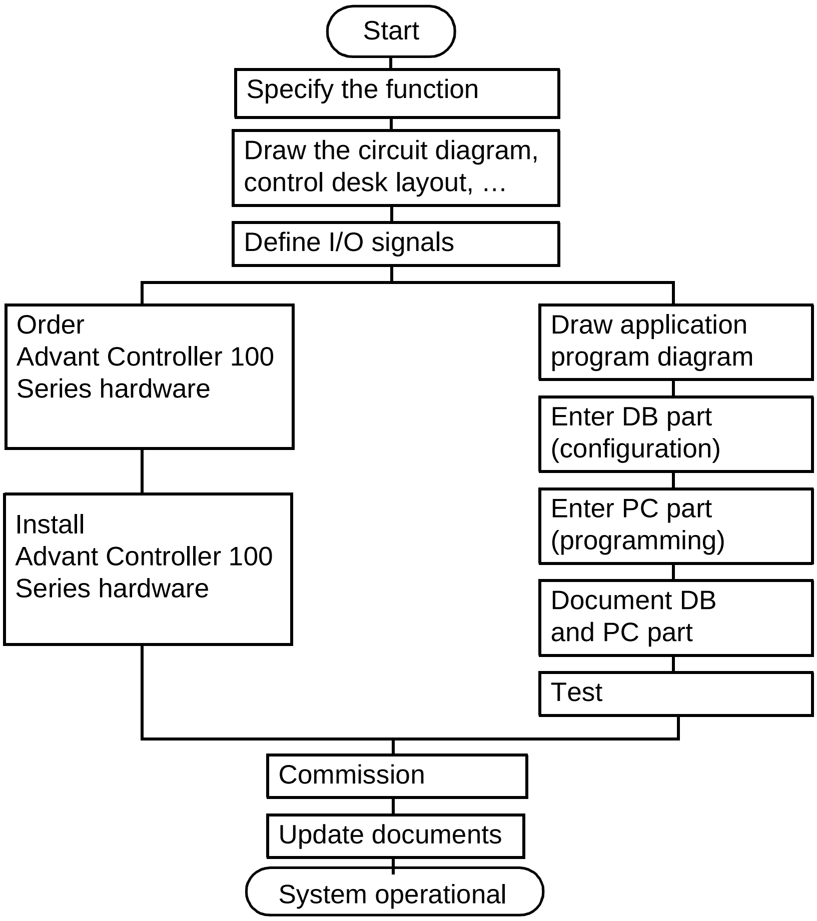

Start
Specify the function
Draw the circuit diagram, control desk layout, …
Define I/O signals
Order Advant Controller 100 Series hardware
Install Advant Controller 100 Series hardware
Draw application program diagram
Enter DB part (configuration)
Enter PC part (programming)
Document DB and PC part
Test
Commission
Update documents
System operational

Figure 2-1. Design of an Advant Controller 100 Series Installation (Flow Diagram)

2.2 Setup of Projects and Nodes on Advant Station 100 Series ES

Before you configure and program Advant Engineering components, you have to do some preparations.
Create at least one project and, for every Advant Controller 400 Series station and Advant Controller 100 Series station, create a node by using the Application Builder.
You may freely choose the division into projects. In one project, you can create several nodes.

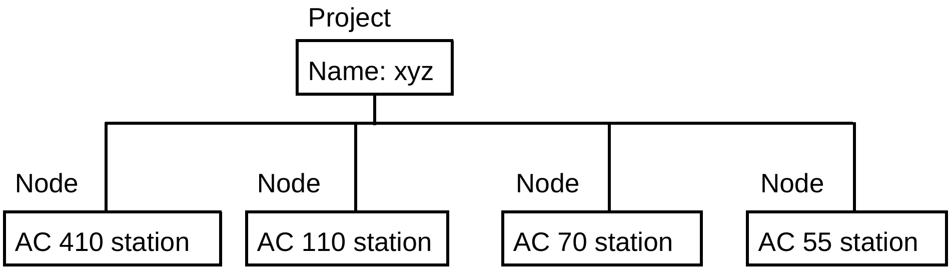
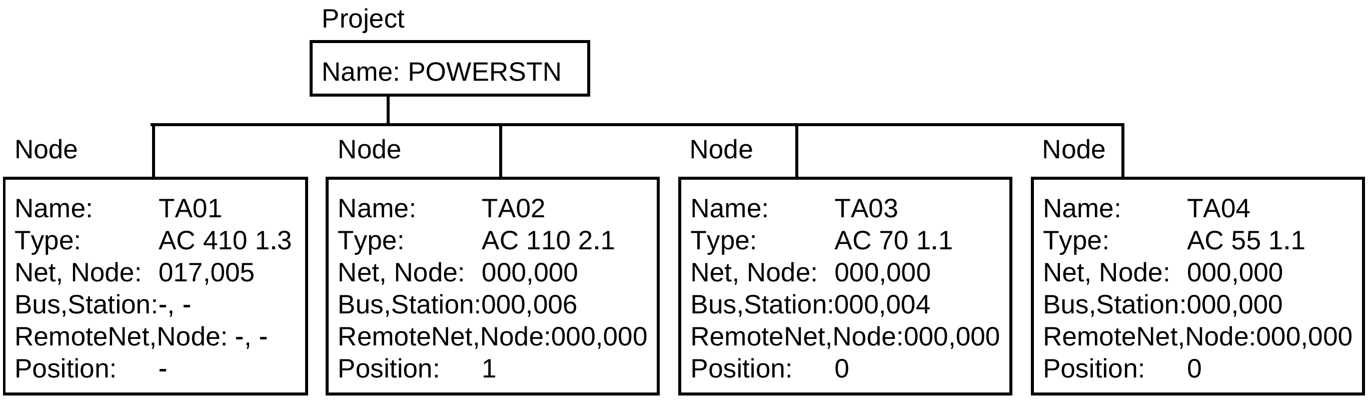


Figure 2-2. Project/Node Structure on Advant Station 100 Series ES, Example

In the Application Builder define the project by name and the nodes by name, type, and address.



Name: TA01
Type: AC 410 1.3
Net, Node: 017,005
Bus,Station:-, -
RemoteNet,Node: -, -
Position: -

Name: TA02
Type: AC 110 2.1
Net, Node: 000,000
Bus,Station:000,006
RemoteNet,Node:000,000
Position: 1

Name: TA03
Type: AC 70 1.1
Net, Node: 000,000
Bus,Station:000,004
RemoteNet,Node:000,000
Position: 0

Name: TA04
Type: AC 55 1.1
Net, Node: 000,000
Bus,Station:000,000
RemoteNet,Node:000,000
Position: 0

Figure 2-3. Project and Node Definition on AS 100 Series ES, Example

In case of Advant Controller 400 Series bus, station, remote net and remote node numbers and position have no meaning.

In case of AC 110, 160, 70, and derivatives, if the controller is connected to an Advant Fieldbus 100 bus, the station number of the node is the controller’s station number on the Advant Fieldbus 100 bus. The station number is the number of the rotary (thumbwheel) switch in the basic station. The station number should be set to zero if there is no Advant Fieldbus 100 bus used. If you want to connect an engineering station via an AC 400 Series station, net number, node number, and bus number of the node are used for connecting.

In case of AC 160 the position number can be 3, 4, or 5, describing of the physical position of the corresponding CPU module. In case of AC 110 the position number is 1.
In case of AC 55 net number, node number, bus number, and station number have no meaning. The station number and position number should be set to zero.
Remote Net, Remote Node represent RCOM address of the node if applicable.

2.3 Directory Structure and Library Handling

Release Directory

At the installation of AMPL Control Configuration, its parts (Application Builder, Function Chart Builder, and Bus Configuration Builder) are stored in their specific release directories. For example, FCB *4.6/0 is stored in the directory C:\AdvaBNT\Fcb\46r0.

The AMPL PC and DB Element Libraries are installed in the ..\Bin32\db_lib and ..\Bin32\pc_lib subdirectories of the FCB release directory. These are the element libraries that you normally use at programming and configuring for new and current projects.

Project Directory

At the creation of a project, the Application Builder creates its directory structure under the project root. For the project POWERSTN it creates, for example, D:\Proj\POWERSTN.

At the creation of a node (and type circuit and circuit), the Application Builder creates its directory structure under the project directory. For the node TA02 it creates, for example, D:\Proj\POWERSTN\Nodes\TA02.

Node-specific Element Libraries

The Function Chart Builder supports also node-specific element libraries. Typical cases where certain nodes may need element libraries that are different from those installed in subdirectories of the FCB release directory:

- You have to program both standard Advant Controllers and their special modified versions.
- You retrieve an application from an Advant Controller 100 Series target together with the libraries. The latter are an earlier release that is not installed on your engineering station.

The node-specific element libraries can be copied (exported/imported) by appropriate APB commands or retrieved from a node supporting this function. Node-specific element library files are stored in the \LIBDATA*release*_*\BIN32\DB_LIB and \PC_LIB subdirectories of the node directory, for example, in D:\Proj\POWERSTN\Nodes\TA02\LIBDATA\45r2\BIN32\DB_LIB and \PC_LIB.

The Function Chart Builder takes the libraries at its start-up into account.
Then it works with the node-specific element libraries, if there are any, instead of the standard libraries.

You can let the Function Chart Builder check the availability of the libraries, rebuild the list of available PC and DB elements, and take them anew into account also during an FCB session. This can be necessary, for example, if you are working on a node with the Function Chart Builder and you change the options, hence the element libraries, for this node with the Application Builder. Or you create a new DB element type with the PROFIBUS Library Editor.

2.4 Connection of Advant Station 100 Series ES to a Controller Station

2.4.1 Examples of Connection

Point-to-point Connection

The point-to-point connection between the engineering station and an Advant Controller 100 Series system is made by a standard V.24 interface at 9600 baud

To establish the V.24 connection between the engineering station and the controller, plug the connection cable into the V.24 connector of the engineering station and the corresponding connector of the controller as described in the user's guide.

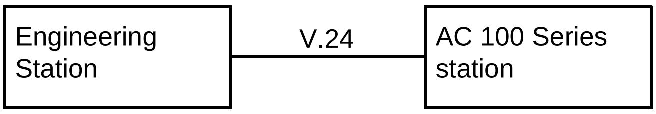


Figure 2-4. Direct Connection of AS 100ES to Advant Controller 100 Series

Routing via AF 100

Advant Controller 400 Series controllers, AC 110, 160, 70, and derivatives, and stations with AF 100 interface installation support the routing function for any remote station via AF 100.

So any AC 110, 160, 70, or derivative station can be reached by the engineering station via Advant Controller 400 Series or another AC 110, 160, 70, and derivative connected to a common Advant Fieldbus 100.

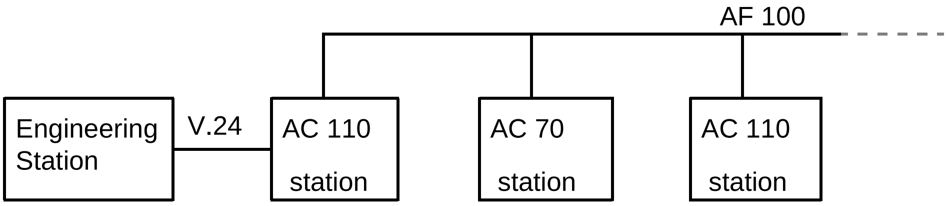


Figure 2-5. Routing from AS 100ES to AC 110 and 70 via AF 100, Example

Routing via RCOM

Advant Controller 400 Series controllers support the routing function for any remote station via RCOM.

So the engineering station connected to an Advant Controller 400 Series station can reach any AC 110, 160, 55, and derivative station via RCOM connecting the Advant Controller stations.

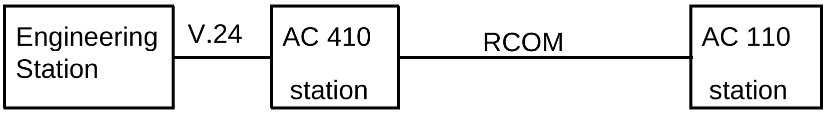


Figure 2-6. Routing from AS 100ES to AC 110 via RCOM, Example

See Section 7.1.2, Connect the Engineering Station to an AC 100 Series Target for more information.

2.4.2 About Server and Client Engineering Stations in Computer Network

Connection of AC 100 Series targets directly or by routing is implemented as Distributed COM application. So engineering station to be connected can be the local one, on which you run the application requesting the connection, such as the Function Chart Builder or the Diagnostics program, but also another remote one, a server, connected to a common computer network. Then your local engineering station ("this PC") is client of the connection service available on the server.

The server can be defined in different forms, like

- Name corresponding to universal naming convention (UNC),
  for example, \\gpa789 or gpa789
- TCP-IP address, for example, 123.456.789.87
- Distributed server name (DSN) like a world-wide-web address,
  for example, www.abb.de or deind.de.

For configuration of the client, the Windows NT system administrator finds the necessary instructions in the *Function Chart Builder User's Guide* (Appendix G, Configuration of AdvaBuild Communicator) and in the on-line help of the Function Chart Builder.

2.4.3 Overview of Remote Access Possibilities

The following table summarizes the routing possibilities depending on the node type of the target and, if applicable, on the intermediate node, to which the server engineering station is connected.

Table 2-1. Routing Possibilities Depending on Node Type

Eng. Station Interface	Routed via 1st Node	AC 110 or 160 Target	AC 70 Target	AC 55 Target
COMx	No routing; no intermediate node.	+	+	+
COMx	AC 110 or 160	AF100, AF100+AF100	AF100, AF100+AF100	-
COMx	AC 70	AF100	AF100	-
COMx	AC 55	-	-	-
X3	AC 400 Series, then AC 110 or 160 as applicable	AF100, AF100+AF100, MB300+AF100, MB300+AF100+AF100, RCOM net, MB300+RCOM net	AF100, AF100+AF100, MB300+AF100, MB300+AF100+AF100	RCOM net, MB300+RCOM net
AF 100 *	No routing; no intermediate node.	+	+	-
Other, such as PCMCIA, ISA, …	No routing; no intermediate node.	-	-	-

Legend:

*	The engineering station has to be additionally equipped with hardware and software necessary for AF 100 interface, such as the CI526 communication interface and AF 100 Interface software installed as a separate component or as part of, for example, AF100 OPC.
+	Direct connection is supported.
-	No routing is supported.
1bus	Routing path is engineering station --> comm. port --> 1st intermediate node --> *1bus* --> target.
1bus+2bus	Routing path is engineering station --> comm. port --> 1st intermediate node --> *1bus* --> 2nd intermediate node --> *2bus* --> target.
1bus+2bus+3bus	Routing path is engineering station --> X3 --> 1st intermediate AC 400 Series node --> *1bus* --> last intermediate AC 400 Series node --> *2bus* --> intermediate AC 110 or 160 node --> *3bus* --> target.
Comma	Separates possible routing paths.

NOTE

Routing possibilities of AC 80 are typically the same as those of AC 70.

2.4.4 Conditions of Successful Connection

1. The connection service is available either on the local engineering station or on a server connected to a common computer network, which you intend to use.
2. If an engineering board is involved in the routing, the On-line Builder must be installed on the pertaining engineering station and the engineering board must be configured appropriately by the Engineering Board Setup program.
3. The engineering station(s), the involved nodes along the routing path, and the target are cabled appropriately.
4. Applications, loaded to the nodes along the routing path, are configured to support the communication via MB 300, AF 100, and RCOM net, as applicable.

2.4.5 Preparation of Connection

Preparation of Engineering Station

If you want to use the connection service running on a remote engineering station, it has to be accessible via computer network from your local engineering station.

See *Function Chart Builder User's Guide*, Appendix G, Configuration of AdvaBuild Communicator for configuration of engineering station - target communication on server.

Preparation of Links

In case of point-to-point RS 232C connection do the following:

> Connect one end of an appropriate cable to the connector, generally denoted by CH1 or CHANNEL 1, of the AC 100 Series target and the other end to a free connector COM1, COM2, … of serial interface of the engineering station.

In case of connection by routing via other nodes and if the first node in the routing path is an AC 100 Series node, do the following:

1. Connect one end of an appropriate cable to the connector, generally denoted by CH1 or CHANNEL 1, of the first AC 100 Series node in the routing path and the other end to a free connector COM1, COM2, … of serial interface of the engineering station.
2. Be sure the nodes up to the target are cabled appropriately to realize the AF 100 communication paths.

In case of connection by routing via other nodes and if the first node in the routing path is an AC 400 Series node, do the following:

1. Be sure the appropriate AS 100 Series Engineering Board is installed correctly in the engineering station and configured appropriately by the Engineering Board Setup program. Choose the latter in the **Start | Advant | Engineering | Utilities** menu.
2. Connect one end of an appropriate cable to the connector, generally denoted by SERVICE, of the first AC 400 Series node in the routing path and the other end to the connector X3 of the AS 100 Series Engineering Board of the engineering station.

3. Be sure the nodes up to the target are cabled appropriately to realize the necessary MB 300, AF 100, and RCOM communication paths.

In case of direct AF 100 connection via the AF 100 interface of the engineering station, do the following:

1. Be sure the AF 100 Interface, that is, the CI526 communication interface module and the AF 100 Interface software, is installed correctly in the engineering station.
2. Be sure the target is cabled appropriately to realize the AF 100 communication.

Preparation of Intermediate Nodes

In case of connection by routing via other nodes be sure appropriate application programs are loaded to the nodes along the routing path that configure them to support the communication via MB 300, AF 100, and RCOM net and to manifest the appropriate addresses.

Chapter 3 User Application Program

3.1 Overview

The user application program is the software you produce using predefined functions in the controller.
The user application program for Advant Controller 100 Series consists of the following parts:

- Configuration part containing the DB elements of the related node
- Program part containing the PC elements of the related PC program.

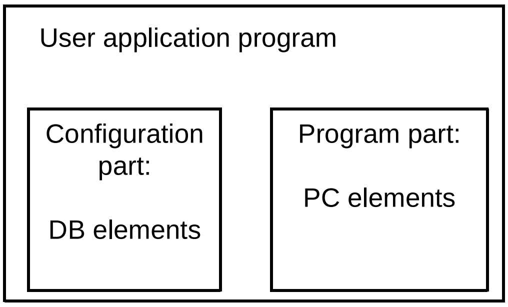


Figure 3-1. User Application Program for Advant Controller 100 Series

3.2 Generation and Storage

Application Program in Source Code Form

You can generate and store a user application program for the controller system in the engineering station.

The common way to create or modify a user application program for an controller system is to use the Function Chart Builder (FCB) in the engineering station.
Another option is to write AMPL source code using a text editor.

You can store a user application program for the controller in two forms:

- Internal data representation of the Function Chart Builder (ODB format), which can be read generally only with FCB of the same release and revision as the one that stored it.
- AMPL source code (ASCII data file), compatible between FCB releases.

Using the FCB command **File | Generate Source**, you can generate AMPL source code from the FCB's internal data representation. Using the FCB command **File | Backtranslate Source**, you can convert the AMPL source code into the FCB's internal data representation.

For information about the general design of AMPL source code, see Chapter 2, Source Code Design, in the *ABB Master Source Code Handling User's Manual*.

Application Program in Target Code Form

After you generate a user application program, the target code of the user application must be generated and then downloaded to the controller station.

You can generate target code for Advant Controller 100 Series only from the FCB's internal data representation by the FCB command **File | Generate Target Code**.

For most node types the target code does not contain all the information about the generated user application program. Because of the controller memory restrictions, information concerning symbolic names, function modules, slave module, and so on, is generally not stored within the target code.

For node types explicitly supporting application retrieval, such as AC 80 2.0, functions are available to regenerate the directories, element libraries, and so on, of a node and upload the target code from an AC 100 Series target and translate the target code back to internal data representation for analysis, editing, and testing with the Function Chart Builder.

For application retrieval, you can use either the **File | Retrieve Application** APB command or the **Target | Upload** and **File | Backtranslate Target Code** FCB commands.

The application can only be retrieved if you work with the same Function Chart Builder version as the one with which its target code was generated. The scope of application retrieval depends on the information included in the target code stored on the target, that is, on the options that were selected at target code generation by the **File | Generate Target Code** FCB command.

For information about APB and FCB, see *Application Builder User's Guide* and *Function Chart Builder User's Guide*.

3.3 Application Protection

The Function Chart Builder offers password protection from manipulations on objects, that is, nodes, circuits, and type circuits, by unauthorized persons. In fact, the access to the *.ODB file containing the internal data representation of the object you are working on is protected by the ODB password.

After creation of a new node, type circuit, or circuit, the initial ODB password is empty and the *.ODB file of the object is not protected.

You can define an ODB password by the **File | Set Password** command.
The Function Chart Builder opens the **Set Password in ODB** dialog box.
Enter here the password consisting of 8 to 50 printable characters, it is case sensitive.

CAUTION

Use password protection only if it is reasonable. Be aware that there is no "back door" or "universal password" if you forget the password. **File | Initialize** does not clear the password. On the other side, files other than *.ODB, for example, source code files, are not protected.

After that, each time you start the Function Chart Builder on this object, it prompts you to enter the password associated with the object.

- If you enter the correct password, the Function Chart Builder starts fully up, opens the *.ODB window, and offers its commands. You can work on the object.

- If you enter an incorrect password, the Function Chart Builder cancels the dialog and exits automatically. You can neither see the application program (PC and DB elements, symbols, terminals), nor change it, just try to enter the correct password again.

You can change or clear the ODB password by the File | Set Password command.
In these cases you have to enter at first the old password.
You clear the password by the empty string as new password:

The access to some Advant Controller 100 Series targets, such as AC 110 2.3, can also be protected by password. See Section 7.2, Target Access Protection. Do not mix up the ODB password with the target password.

3.4 Backup and Restore

You can back up an application program which is running on the controller and is stored in the RAM or the flash PROM of the controller only in case of node types explicitly supporting application retrieval, such as AC 80 2.0.

CAUTION

If you lose your data on the engineering station and there is no backup,
you can neither document the actual program, nor make on-line changes.
You must re-create your entire program on the engineering station.

3.4.1 Save and Backup

It is very strongly recommended to save the actual user data of the engineering station on a backup medium, for example, floppy disks or network drives.

It is a good practice to do the following:

- Save the user data frequently during the creation of the application program.
- Save the user data when the creation of the application program on the engineering station is finished.
- Save the user data after every change within a “finished” application program.

To save and back up use the FCB command **File | Save** and the Application Builder commands **File | Copy** (first of all **File | Copy Project**), possibly also some packing utility.

Store the backup medium in a safe place.

3.4.2 Restore

To restore use the Application Builder commands **File | Copy** (first of all **File | Copy Project**), possibly also some unpacking utility on the engineering station.

Chapter 4 Configuration

4.1 General

The controller station consists of a basic unit or CPU module, a power supply, and modules for process I/O and/or communication interfaces (see Figure 4-1). Or the basic unit itself contains also I/O channels.

The configuration part of a user application program contains the configuration elements of the controller system:

- Hardware description
- Common data description
- Data exchange description.

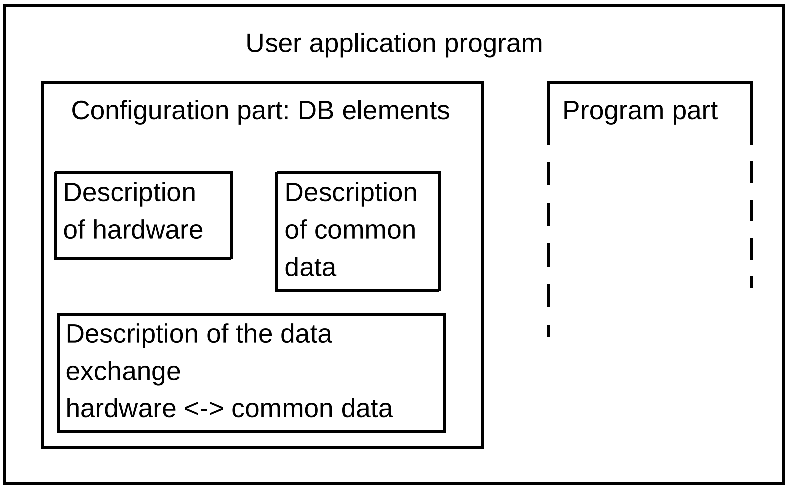

User application program
Configuration part: DB elements
Program part
Description of hardware
Description of common data
Description of the data exchange hardware <-> common data

Figure 4-1. Configuration Part Within User Application Program

4.1.1 Definition

When you configure a controller station, you describe the hardware of the controller station, the common data area, and the data exchange by means of DB elements.

See *Data Base Elements Advant Controller 100 Series Reference Manual*,
Data Base Elements Advant Controller 160 Reference Manual,
Data Base Elements Advant Controller 70 Reference Manual, and
Data Base Elements Advant Controller 55 Reference Manual for detailed description of each DB element.

4.1.2 Hardware of Advant Controller 100 Series

An Advant Controller 160 station consists of one or several CPU modules, a power supply and modules for process I/O and/or communication interfaces, such as digital input modules or interface to the Advant Fieldbus 100 or to the MasterFieldbus (MFB).
The Advant Controller 160 basic (sub)station contains the CPUs. Extensions to the Advant Controller 160 basic station can be housed in I/O (sub)station.

An Advant Controller 110 station consists of a CPU module, a power supply and modules for process I/O and/or communication interfaces, such as digital input modules or interface to the Advant Fieldbus 100 or to the MasterFieldbus (MFB).
The Advant Controller 110 basic (sub)station contains the CPU. Extensions to the Advant Controller 110 basic station can be housed in I/O (sub)station.

An Advant Controller 70 station consists of a CPU module and modules for process I/O, which can be housed in several I/O clusters. An Advant Controller 80 has additionally several built-in interfaces and Drives specific modules.

An Advant Controller 55 station consists of a basic unit and optionally several expansion units.

See *Advant Controller 160 User's Guide*, *Advant Controller 110 User's Guide*, *Advant Controller 70 User's Guide*, *Advant Controller 55 User's Guide*, and so on, for detailed hardware descriptions.

4.2 Configuration by DB Elements

DB elements in a controller system describe the following items:

- The hardware of the controller system:
  - CPU module, basic unit, expansion unit as applicable depending on the controller
  - Digital input, digital output, analog input, analog output, and so on
  - Communication interfaces, like MVICHAN or connection to AF 100 as applicable for the controller
- Common data elements:
  - DAT(B), DAT(I), DAT(IL), DAT(R), DAT(II)
  - PARDAT(B), and so on, PARDAT(NB), and so on.
  - DAT_AI, DAT_AO, DAT_DAT, DAT_DI, and DAT_DO elements carrying HMI-relevant properties of signals
- Event handling
- Connection between the hardware and the common data elements as applicable depending on the controller: DataSet Peripheral DSP, MVI Data Block MVB
- In case of AC 110, 160, and derivatives also extended DB elements covering a set of the previous items and HMI functions: AIS, DIS, AOS, DOS, MB, MBS, MI, MIL, and MR
- In case of AC 110 *2.2, AC 110 *2.3, AC 160 >=*1.1 also DB elements PB, PBS, and other specific elements describing PROFIBUS-DP and PROFIBUS devices.

For information about connections between DB and PC elements, refer to Section 5.5.2, Connection of Elements.

4.3 Call Names of DB Elements

When you create DB elements on an engineering station, you select them by their call names. Call names of DB elements describing modules and channels correspond to the identifier of the related units (for example, CT482, AIS481) or modules (for example, AI610). For other elements, the call name equals the element name or is a short mnemonic of the element name.

Table 4-1. Examples of Call Names Advant Controller 110

DB Element Type	Call Name(s)
Processor Module	PM632, PM633
Analog Input	AI610, AI620, AI625, AI630, AI635
Analog Output	AO610, AO650
Digital Input	DI610, DI620, DI621, DI622, DI635, DI636, DI650, DI651, DI652
Digital Output	DO610, DO620, DO625, DO630
Pulse Counter	DP620
Communication Interface	CI610, CI615, CI625, CI626, CI627, CI670, CI671
Submodule Carrier	SC610
Boolean Data	DAT(B), PARDAT(B), PARDAT(NB)
Integer Data	DAT(I), PARDAT(I), PARDAT(NI)
Integer Data, Double	DAT(II)
Integer Long Data	DAT(IL), PARDAT(IL)
Real Data	DAT(R), PARDAT(R)
DataSet Peripheral	DSP
Multi Vendor Configuration	MVICHAN, MVINODE, MVB
EventSet, Event MVBs	EVS(S), EVS(T), AIMVB, DIMVB
Analog Input Calculated	AIC
Digital Input Calculated	DIC
TYRAK_L Element	TYRAK_L
Extended DB Elements, I/O	AIS, DIS, AOS, DOS
Extended DB Elements, memory	MB, MBS, MI, MIL, MR
HMI Elements	DAT_AI, DAT_AO, DAT_DAT, DAT_DI, DAT_DO
PROFIBUS-DP Elements	PB, PBS
AF 100 Station Element	AF100S

Table 4-2. Examples of Call Names Advant Controller 70

DB Element Type	Call Name(s)
Processor Module	PM810
Analog Input	AI810, AI830, AI835
Analog Output	AO810
Digital Input	DI810, DI820
Digital Output	DO810, DO820
Boolean Data	DAT(B), PARDAT(B)
Integer Data	DAT(I), PARDAT(I)
Integer Long Data	DAT(IL), PARDAT(IL)
Real Data	DAT(R), PARDAT(R)
DataSet Peripheral	DSP
EventSet	EVS(S)
Analog Input Calculated	AIC
Digital Input Calculated	DIC

Table 4-3. Examples of Call Names Advant Controller 55

DB Element Type	Call Name(s)
Basic Unit	CT481, CT482
Analog Input	AIS481, AIS482
Analog Output	AOS481, AOS482
Digital Input	DIS481, DIS482
Digital Output	DOS481, DOS482
Expansion Unit	CX473, CX475
Analog Input	AIS473, AIS475
Analog Output	AOS473, AOS475
Digital Input	DIS473, DIS475
Digital Output	DOS473, DOS475
Boolean Data	DAT(B), PARDAT(B)
Integer Data	DAT(I), PARDAT(I)
Integer Long Data	DAT(IL), PARDAT(IL)

Table 4-3. Examples of Call Names Advant Controller 55 (Continued)

DB Element Type	Call Name(s)
Real Data	DAT(R), PARDAT(R)
Multi Vendor Configuration	MVICHAN, MVINODE, MVB
EventSet	EVS(S)
Analog Input Calculated	AIC
Digital Input Calculated	DIC

4.4 Graphic Symbols of DB Elements

On the engineering station, the data base elements are visually represented by graphic symbols (“boxes”). For large elements, the symbols is sub-divided into a number of segments (“parts”).

When an element is displayed on the screen of the engineering station, the header and one expanded segment are shown. Other segments may be shown in compressed form.

The segments of the symbol (box) have one terminal for each data item. Configuration data are shown on the left-hand side and signals are shown on the right-hand side of the element.
The names inside the symbol are terminal names that identify the data items. These names are used, for example, when PC elements are connected to data base elements.

4.5 Item Designation of DB Elements

All DB elements have a unique item designation within the node.

The item designation consists of a first part, which describes the kind of DB element, and a concatenated number, which describes, for example, the number of a module within the same type.

Channel DB elements also have a second running number, the channel number, so the relation between module and channel DB elements is visible.

Table 4-4. Examples of Item Designation

DB Element	Item Designation
Module element DI its channels:	DI1 DIS1.1 DIS1.2 ...
DAT(B) element DAT(I) element DAT(B) element ...	DAT1 DAT2 DAT3 ...

4.6 Names of DB Elements

All DB elements have a NAME terminal. The default name is the item designation. A user-defined name can be given to this terminal. Use this name as a reference in PC programing when connecting to DB elements.

The name can be up to 20 character long, except the following:
Names of DAT(II), PARDAT(II) elements can be up to 19 character long.
Names of DAT, DSP, and MVB can be up to 12 character long.
Names of specific PROFIBUS-DP elements can be up to 34 character long.

The name can contain most characters with the exception of a space.
You can use a name that might be interpreted as an item designation only if it is the exact item designation of the element.

4.7 Configuration of DB Elements Describing the Hardware

4.7.1 Overview

Every CPU module or unit is described by means of at least one DB element.
A module DB element or unit DB element represents all specific properties of a module or unit, respectively. Its terminals represent the following:

- Module position or unit address
- Module-specific parameters
- Module error.

Most of the module or unit DB elements have subordinated DB elements representing channels of the related module or unit, for example, a module DB element of type DI or CT482 has several subordinated channel DB elements. Channel terminals represent the following:

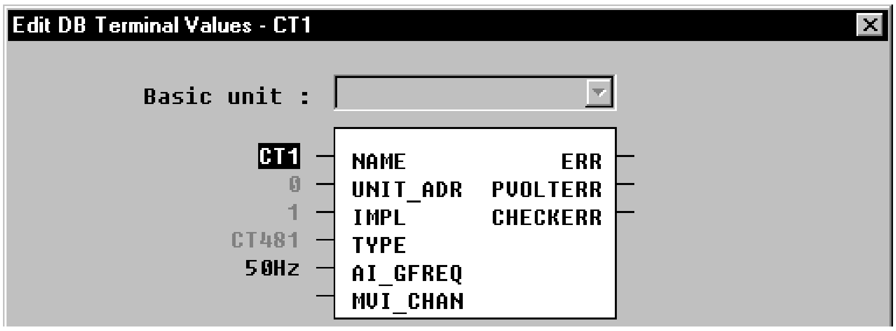
- Channel-specific parameters
- Channel value, that is, the process value
- Channel error.

When you create a module or unit element on the engineering station, the subordinated channel elements are created automatically.

When you create a remote I/O elements (S800 I/O), extended DB elements, and AF100S element on the engineering station, pertaining DSPs are created automatically. The latter can be made visible on the engineering station, then they are denoted by the attributes R, X, and S, respectively.

4.7.2 Configuration of Unit or Module DB Elements

See an example of the engineering station view of a basic unit DB element (CT481 for AC 55):



Edit DB Terminal Values - CT1

Basic unit :

CT1 — NAME
0 — UNIT_ADR
1 — IMPL
CT481 — TYPE
50Hz — AI_GFREQ
— MVI_CHAN

ERR —
PVOLTERR —
CHECKERR —

Figure 4-2. Engineering Station View of Basic Unit (CT481, Example)

Input terminals, that is, terminals on the left-hand side of a basic or expansion unit or module element, can be:

- NAME a unique name for this DB element
- Terminals representing the address of the module:
  - BUS
  - STATION
  - POSITION.
- UNIT_ADR fixed to 0 in case of the basic unit, 1 to 4 for expansion units
- IMPL IMPL = 0 means the unit or module (and its channels) will not be initialized and activated at the next system start-up
- TYPE The unit or module ID, given by the call name of the element
- AI_GFREQ the filter grid frequency 50 or 60 Hz for the AI channels
- MVI_CHAN a reference to a separately created MVI channel if any
- Terminals representing parameters valid for the whole module, for example,
  - CONV_PAR
  - GRIDFREQ.

Output terminals, that is, terminals on the right-hand side of a basic unit or module element, can be the following:

- ERR error signal to indicate a module hardware or configuration error
- PVOLTERR error on analog or digital process voltage
- CHECKERR checksum error of application PROM
- DIAG state of module (in case of AC 70 and S800 I/O).
- STATUS1, 2, and 3 state of module (in case of AC 160).

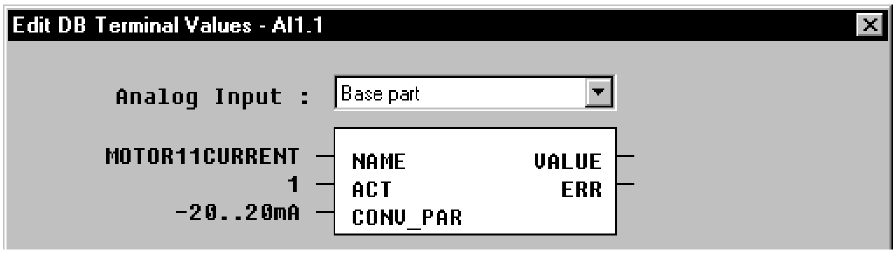
NOTE

> The ERR terminal of DI, DO, AI, and AO channel DB elements does not indicate missing process voltage if the basic unit DB elements has a special terminal for this purpose.

AC 160 supports redundant CPUs. The DB element of its PM645 processor module has also a so called redundancy part.
For input terminals (NO, CPU, etc.) and output terminals in the redundancy part (TR_GRANT and SWITCH), see *Advant Controller 160 User's Guide*.

4.7.3 Channel DB Elements

Every channel of a unit or module is represented by at least one DB element, the channel element. See an example of the engineering station view of a channel:



Edit DB Terminal Values - AI1.1

Analog Input : Base part

MOTOR11CURRENT — NAME VALUE —
1 — ACT ERR —
-20..20mA — CONV_PAR

Figure 4-3. Engineering Station View of Channel, Base Part (AI620, Example)

Terminals on the left-hand side of a channel element are:

- NAME Channel elements can be given a unique user name allowing an alternative identifying method within application programs. The default user name for the element is the item designation.
- ACT Can be modified to determine the initialization of a channel. ACT = 0 means the related channel will not be initialized and activated at the next module initialization.
- CONV_PAR Terminal representing signal range for the channel.

Terminals on the right-hand side of a channel element are signals to be used within the application program, for example, the value of the process data or the error signal to indicate a process data error:

- VALUE
- ERR.

When you create a module element on the engineering station, the related channel elements are created automatically.

Examples

If you create module element AX645 for an AC 160, channels AXIS and AXOS are generated automatically.

If you create module element AI620 for an AC 110, channels AIS are generated automatically.

If you create module element CT481 for an AC 55, channels AIS, AOS, DIS, and DOS are generated automatically.

4.7.4 Hardware Initialization

Modules and their channels, with the parameters described in the related DB elements, are always initialized after start-up of the controller system with application, after downloading an application, and after on-line changes (see Section 11.8, On-line Changes).

The configuration values (parameters) for the I/O modules and expansion units are stored in a memory data area on the controller’s CPU module or basic unit.

4.8 Configuration of DB Elements Describing Common Data

4.8.1 Overview

The common data area of a controller system is represented by the DB elements DAT and PARDAT:

- DAT and PARDAT DB elements represent memory locations in the common data area.
- Some programs use DAT and PARDAT DB elements for inter- control module communication.
- All DAT and PARDAT elements are readable and writable. In the case of AC 110, 160, and derivatives, PARDAT elements are readable and writable also by a local HMI (MMI) connection.
- DAT elements can also be part of DataSet Peripheral (DSP) and MVI Data Block (MVB) elements for data exchange via Advant Fieldbus 100 and RCOM, respectively, as applicable depending on the controller.
- DAT elements are initialized with value = 0.
- PARDAT are DAT elements with initial values given by the INIT_VAL terminal of the PARDAT element.
  PARDAT elements may not be components of DSP or MVB elements.

Table 4-5. DB Elements Representing Common Data Area

Element	Data Type
DAT(B)	32 packed Boolean values
DAT(I)	Integer
DAT(II)	Two integers
DAT(IL)	Integer long
DAT(R)	Real
PARDAT(B)	32 packed Boolean values with initial values
PARDAT(NB)	32 packed Boolean values with initial values, each value named
PARDAT(I)	Integer, with initial value
PARDAT(II)	Two integers, with initial values
PARDAT(NI)	Two integers, with initial values, each value named
PARDAT(IL)	Integer long, with initial value
PARDAT(R)	Real, with initial value

In case of AC 160, DB elements are also available for storing common data of several redundant CPU modules. Examples: MDAT(B), MDAT(I), MDAT(IL), MDAT(R).

4.8.2 Configuration of DAT/PARDAT/DAT_/MDAT Elements

DAT

Every value in the common data area of a CPU module is represented by a DAT(*i*), PARDAT(*i*), DAT_*i*, or M*i* DB element.

You use DAT(*i*), that is, DAT(B), DAT(I), DAT(II), DAT(IL), and DAT(R), elements to represent some common data that have the initial value 0 and are often referenced by a DSP element. DATs referenced by DSP are used to represent data communicated via Advant Fieldbus 100.

PARDAT

You use PARDAT(*i*), that is, PARDAT(B), PARDAT(I), PARDAT(II), PARDAT(IL), PARDAT(NB), PARDAT(NI), and PARDAT(R), elements to represent some common data that serve as parameter of the application.
PARDAT(*i*) elements have a programmable initial value. By the Function Chart Builder you can read the actual values of the PARDAT(*i*) elements from a target and store them as new initial values to tune the application.

DAT_

DAT_*i*, that is, DAT_AI, DAT_AO, DAT_DI, and DAT_DO, elements represent the name of appropriate AI, AO, and so on, channels for HMI and can generate an event.
DAT_DAT elements represent the name of DAT elements for HMI.
You use this elements to define the signal names (to be presented in HMI) in the early phase of configuring the application with the Function Chart Builder.

MDAT

You use M*i*, that is, MB, MI, MIL, and MR, elements to represent some common data, also their names elements for HMI, they can also generate an event.

Common Terminals

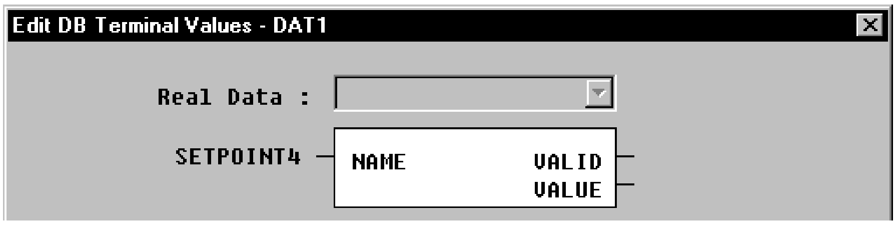


Figure 4-4. Engineering Station View of DAT(R)

Terminals on the left-hand side of a DAT/PARDAT element are:

- NAME DAT and PARDAT elements can be given a unique user name. Use this name for identification within application programs. The default user name for the element is the item designation.
- INIT_VAL An initial value valid at start-up of an application program (only for PARDAT elements).

Terminals on the right-hand side of a DAT/PARDAT element are data to be used within application programs:

- VALUE The value of the data, initialized by the system with:
  - 0, for all DAT elements
  - INIT_VAL, for all PARDAT elements.
- VALID A flag normally used for indicating the validity of the data, initialized by the system with 0, and to be read and written by the application program, or in case of AC 110, 160, and derivatives, by HMI (not supported for PARDAT elements).

NOTES

1. **Contiguous Usage of Item Designation for DAT/PARDAT**

   The DAT/PARDAT instances DAT1/PARDAT1, DAT2/PARDAT2, and so on, occupy fixed addresses of the common data area of the CPU module of AC 110, AC 160, AC 70, and AC 55 depending on their item designation suffix 1, 2, and so on.
   For example, if you configure DAT1 and DAT55, they occupy as much common data area as DAT1, DAT2, and so on up to DAT55.

   To economize common data area of AC 110, AC 160, AC 70, and AC 55, configure DATs/PARDATs starting at DAT1/PARDAT1 and use consecutive designation suffixes. In our example configure DAT1 and DAT2 instead of DAT1 and DAT55.

   These notes do not concern AC 110 2.0, AC 70 1.0, and AC 55 1.0.

2. **About MDAT(*i*) Elements**

   DB elements MDAT(*i*), available for AC 160, are similar to DAT elements. They refer to common data area in the CI626 communication module that is used by for several redundant CPU modules in the same station.

   MDAT(*i*) has the input terminal NAME and the output terminals VALUE and ERR.

   Use the same item designation of MDAT(*i*) to access the same common data from the applications running on different redundant CPU modules.

3. **About PARDAT(NB) and PARDAT(NI) Elements**

   In contrast to DAT(B) and DAT(I), the PARDAT(NB) and PARDAT(NI) elements have channels for their data. So they allow assignment of individual instance names to each binary and integer data, respectively.

4. **About DAT(II) and PARDAT(II) Elements**

   In contrast to DAT(I) and PARDAT(I), the DAT(II) and PARDAT(II) hold two integers, so they need relatively less memory for storing an integer.

5. **Creation of DAT_DAT Elements**

   You can create manually DAT_DAT elements referencing DAT elements, the former conveying the names of the latter to HMI.
   At target code generation, if the **Bus Configuration Database** and **All DSPs Visible in AF100 OPC** options are selected, the Function Chart Builder creates automatically DAT_DAT elements for each DSPi:REFj terminal to convey names of the referenced DATs to HMI. These DAT_DATs—in contrast to those created manually—do not appear in the internal data representation (*.ODB file) of the node, only in the Bus Configuration Database.
   Avoid mixing manual and automatic creation of DAT_DAT elements.

Configuration of DAT and DAT_DAT Elements

See the following figure:

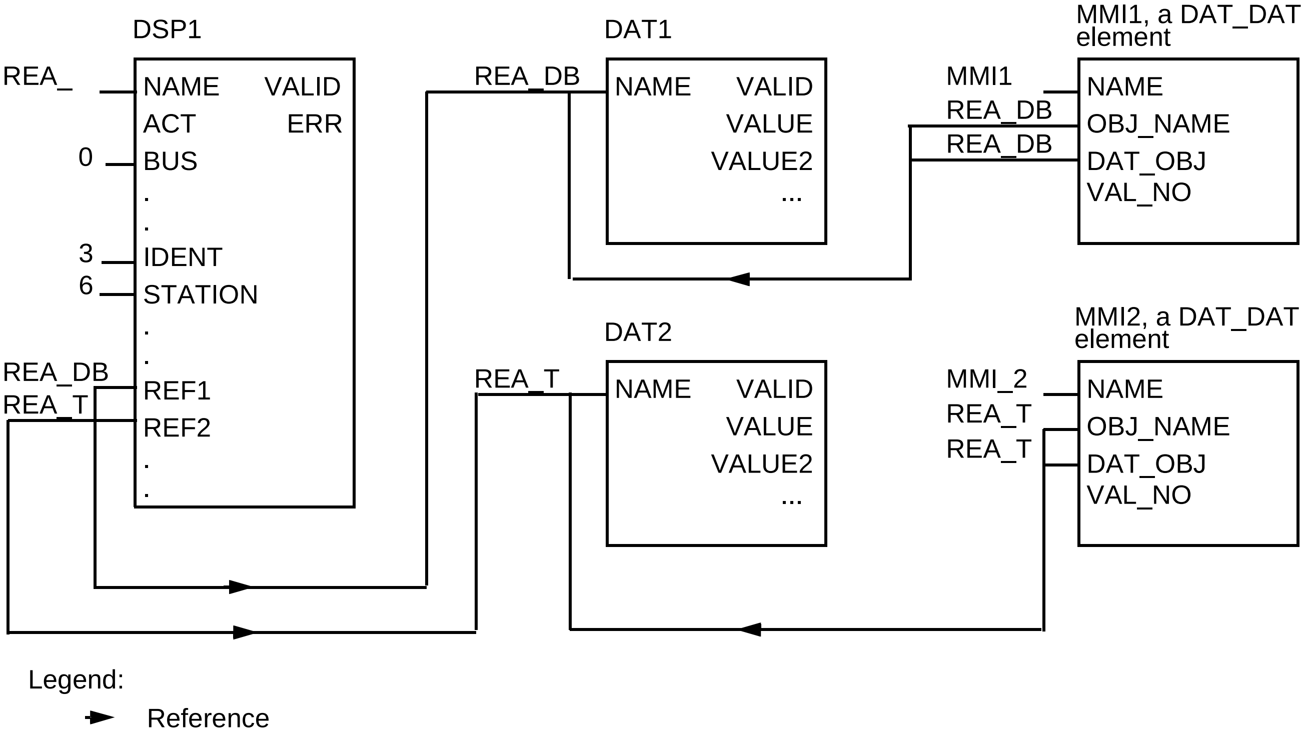


Figure 4-5. DAT_DAT Usage to Convey Names to HMI

The DSP represents a telegram on the AF 100. The DATs define the process data constituting the telegram. The HMI accesses the process data DAT*i*:VALUE via AF 100.

The DAT_DATs reference the DATs and hold the names of the latter as defined on the DAT*i*:NAME terminals. At target code generation the Function Chart Builder creates the DAT_DAT elements in the Bus Configuration Database (and/or BOIL, COIL).
At its configuration the HMI acquires the DAT names there.

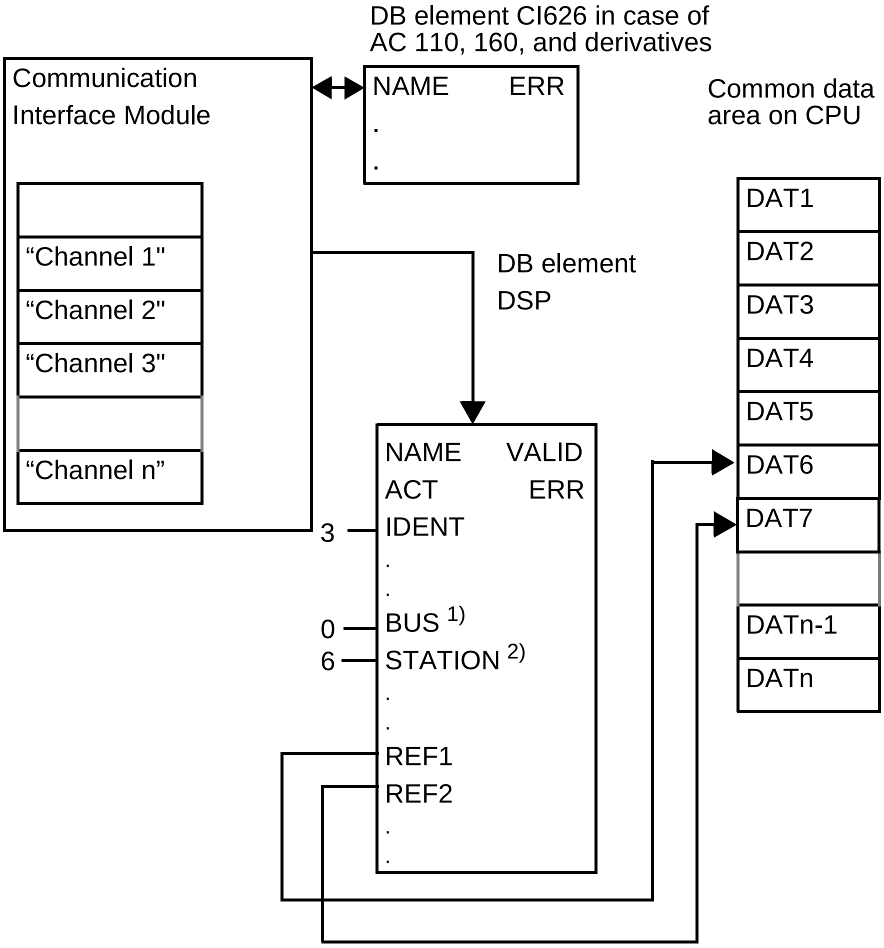
4.9 Communication via Advant Fieldbus 100

4.9.1 Overview

The DataSet Peripheral (DSP) elements are used for the process data communication via Advant Fieldbus 100.
This section is applicable only for Advant Controllers supporting Advant Fieldbus 100.
In the next two figures we show the main elements in both the sending and receiving station necessary to implement communication via Advant Fieldbus 100.

Node of sending station:

Name	**Type**	**Net, Node**	**Bus, Station**
TA01	AC 110, 160, 70, and derivatives	000,000	000,006



1) Terminal value has to be equal to the own (physical) bus address.

2) Terminal value does not necessarily reflect the own physical station address. Using further, “virtual” station numbers makes it possible to have more than 50 (=max IDENT) sending DSPs per station.

Figure 4-6. DataSet Communication, Sending Station

Node of receiving station:

Name	Type	Net, Node	Bus, Station
TA02	AC 110, 160, 70, and derivatives	000,000	000,003

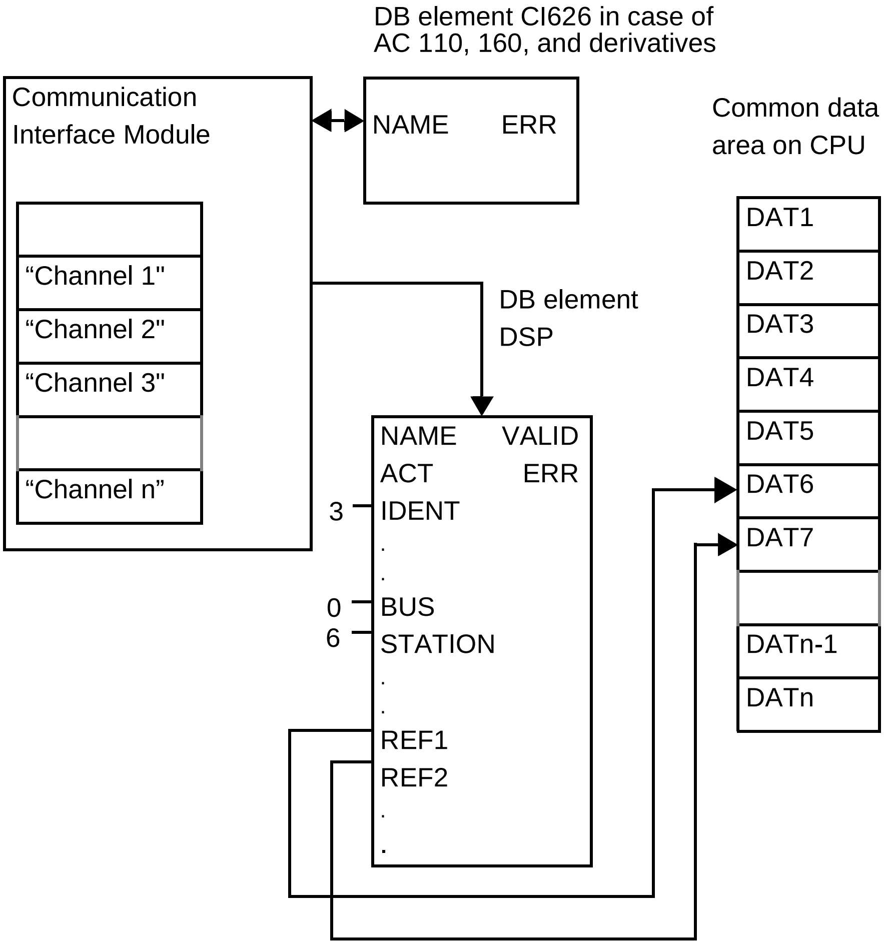


Figure 4-7. DataSet Communication, Receiving Station

IDENT and STATION must be the same at both DSP elements. BUS must define the same bus, though you are not forced to give one bus the same bus number in all controllers.

4.9.2 Configuration of DataSet Peripheral Element

The DataSet Peripheral DB elements represent “channels” on a communication interface and points out the data of the common data area to be transmitted via these “channels”.

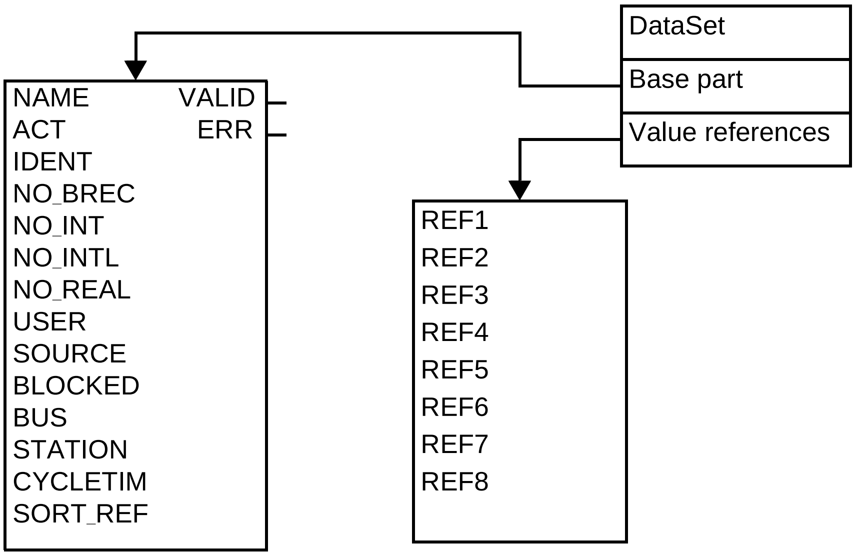


Figure 4-8. Representation of DSP Element, Overview

Special properties of the DataSet Peripheral DB element DSP (see Figure 4-8) are:

ACT	The ACT terminal specifies whether the element is active or not.
IDENT	The IDENT (together with STATION) identifies the DataSet Peripheral. AC 110 and 160 support up to 200 DataSet Peripherals per station. AC 70 supports up to 62 DataSet Peripherals per station. AF 100 allows 1<= IDENT <= 50 as bus address.
USER	The USER terminal must be set to zero for AF 100 DataSet communication.
BLOCKED	The BLOCKED terminal is not used by AC 110, 160, 70, and derivatives. It exists for compatibility reasons only.
BUS	An AC 110 or 160 station can be connected to maximum of two AF 100s, an AC 70 to maximum one AF 100.
STATION	Specifies the value on the STATION terminal of the sending station.
CYCLETIM	The scan factor specifies the transmission interval of the DataSet Peripheral.

When you are creating a DSP element on an engineering station, you can specify the number of packed Boolean data DAT(B), integer data DAT(I), integer long data DAT(IL), and real data DAT(R) as NO_BREC, NO_INT, NO_INTL, and NO_REAL.

This causes the engineering station to automatically create the corresponding DAT elements. The NAME of the DSP element is used as the first part of the DAT name.

The length of a DataSet Peripheral in bytes is calculated by:

length(bytes) = 4 * (NO_BREC + NO_INT + NO_INTL + NO_REAL)

The length of a DataSet Peripheral may not exceed 32 bytes.

It is possible to make references to existing DAT elements when you are creating the DSP element on an engineering station. These references automatically declare the actual numbers (NO_BREC, …) of the DAT elements to which are referenced.

The numbers of DAT elements (NO_BREC, …) are justified if the number of references does not reflect the actual numbers.

For already existing DSP elements, you can delete existing references and add new references.

When you leave the DSP element on the engineering station, the references are stored in the order B, I, IL, R without any empty REFx, if SORT_REF = Yes.

NOTE

If the name of the DSP element is changed later on, the names of the corresponding DAT elements are not automatically updated!

The SOURCE terminal defines the direction of the communication (R=receive, S=send).

4.10 Advant Fieldbus 100 Network Configurations

To exchange data between different AC 110, 160, or 70 stations or with an Advant Controller 400 series process station, you can connect up to 80 nodes via Advant Fieldbus 100.

This section is not applicable for AC 55.

The Advant Fieldbus 100 interface of AC 70 station is realized in the CPU module. The Advant Fieldbus 100 interface of AC 110 and 160 station is realized by the CI626 and CI627 communication interface modules.

For process data exchange via Advant Fieldbus 100, use the DataSet communication.

NOTE

In the following, we use the term DataSet instead of DataSet Peripheral for simplicity.

An AC 110 or 160 node can be connected to maximum two Advant Fieldbus 100.

An AC 70 node can be connected to only one Advant Fieldbus 100.

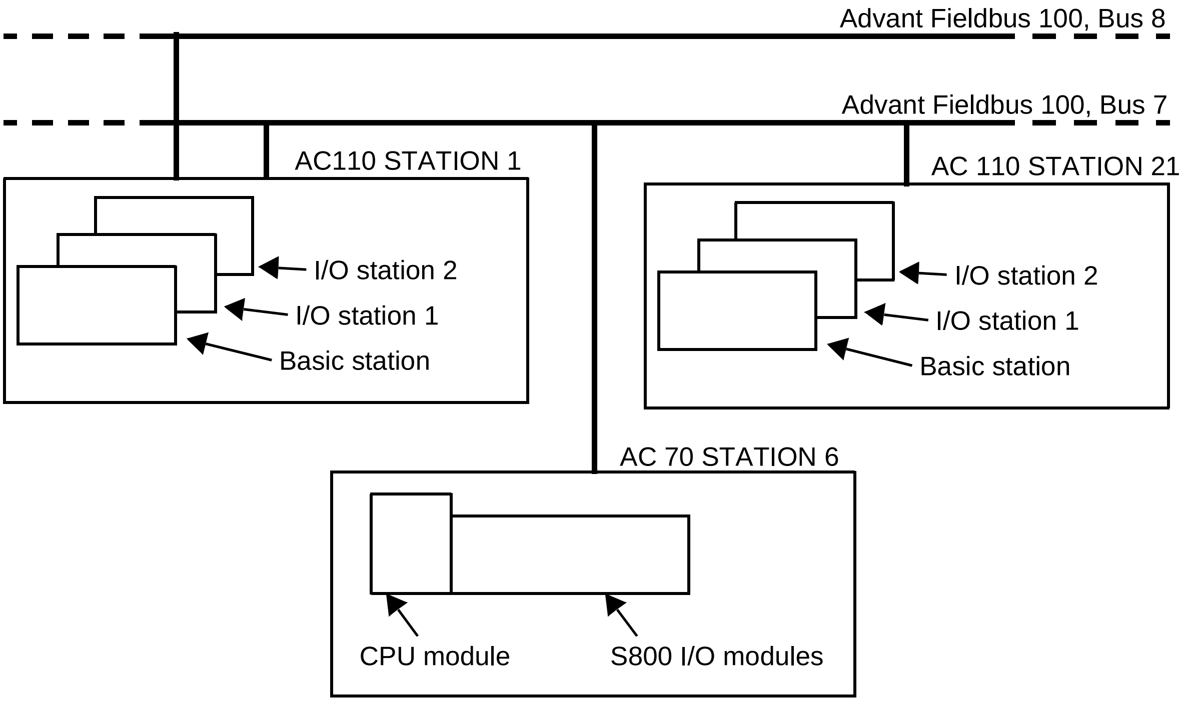


Figure 4-9. AC 110 and 70 Stations Connected via Advant Fieldbus 100

On Advant Fieldbus 100, all transmitted DataSets are identified by IDENT and STATION of the sending node. All nodes connected via the same Advant Fieldbus 100 are capable of receiving every DataSet transmitted on the Advant Fieldbus 100.

The DSP element represents a “channel” of Advant Fieldbus 100 communication.
It points out the data of the common data area to be transmitted via this “channel”.

See Figure 4-9 and Figure 4-10.

Figure 4-9 shows Advant Fieldbus 100, Bus 7 connecting the stations 1, 6, and 21.
Advant Fieldbus 100, Bus 8 illustrates the possibility to connect a second Advant Fieldbus 100 to an AC 110, 160, or derivative.

Figure 4-10 shows the DataSets for the configuration of the connection of stations 1, 6, and 21 to the Advant Fieldbus 100, Bus 7. The DataSet necessary for the configuration of the connection of station 1 to the Advant Fieldbus 100, Bus 8 is not shown.

To receive a DataSet in a node, IDENT and STATION of the receiving DataSet must be the same as IDENT and STATION of the corresponding sending DataSet in the sending node.

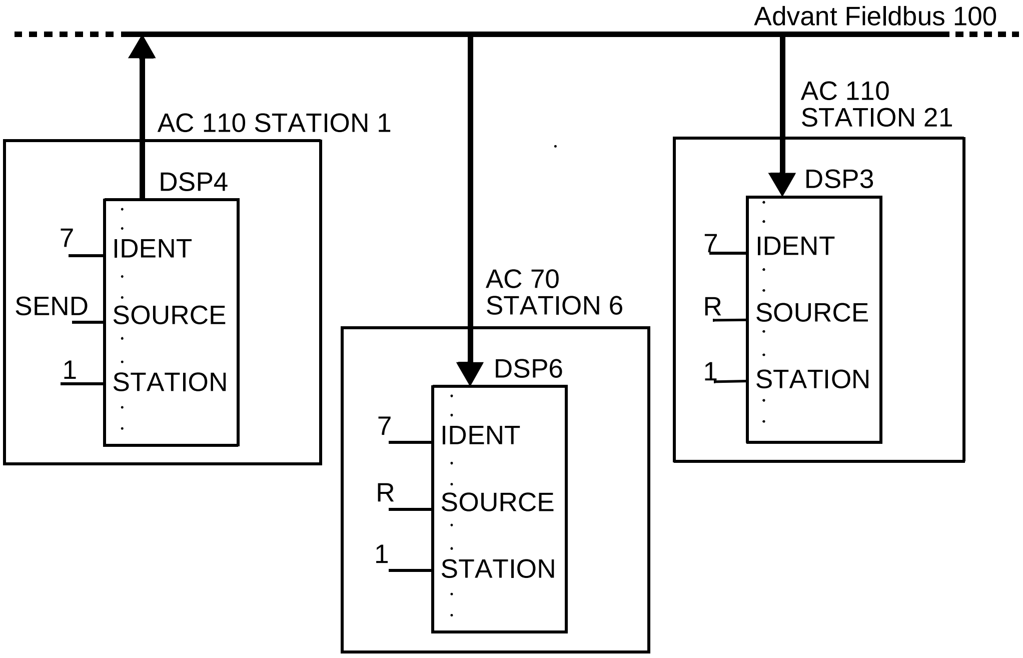


Figure 4-10. DataSet Communication via Advant Fieldbus 100 (Example)

There are three scan factors controlling the transmission of data via AF 100. In the sending AC 110, 160, 70, or derivative station, the cycle time of the PCPGM determines the scan factor for the transfer from the common data area to the communication module. The CYCLETIM of the DataSet determines the scan factor for the transfer from the communication module to the AF 100.
The cycle time of the PCPGM in the receiving AC 110, 160, 70, or derivative station determines the scan factor for the transfer from the CI626 module to the common data area.

The cycle time of the PCPGM executable unit must, therefore, be chosen according to the fastest CYCLETIM of a DataSet, in order to write or read the DataSets' contents as often to or from the coupler module, as it is really transferred between the couplers. In order to save CPU load there is an automatic mechanism which controls writing and reading of contents of slower DataSets in a way that they are not written or read in every cycle of PCPGM executable unit, but only if necessary according to their CYCLETIM.

The CYCLETIM of the DataSet in the receiving AC 110, 160, 70, or derivative station is for supervision of the transfer cycles only.

4.11 Communication via RCOM

AC 110, 160, 55, and derivatives provide remote communication over multidrop, point-to-point, and dial-up connections using the protocols RCOM (Remote COMmunication) and RCOM +.

This section is applicable only for Advant Controllers supporting RCOM.

4.11.1 Overview

For the implementation of data exchange via RCOM and RCOM+ different hardware modules and software options are required in the AC 110, 160, 55, and derivative stations.

In case of AC 110 and 160 the hardware comprises the SC610 carrier module and the CI 532 asynchronous communication submodule (Multi Vendor Interface). One channel is realized in the CPU module itself.

In case of AC 55 the necessary hardware is realized in the CPU module itself.

The OPT2: RCOM option system software has to be available on the controller.

To configure RCOM communication, MVI Data Block (MVB), MVI Channel (MVICHAN), and MVI Node (MVINODE) DB elements are used.

For more and detailed information see *RCOM Advant Controller 110 User's Guide* and *RCOM Advant Controller 55 User's Guide*.

4.11.2 Configuration of MVI Data Block Element

The MVI Data Block (MVB) elements represent “commands” or “data” packed in a collection of DATs for the Multi Vendor Interface protocol handler.

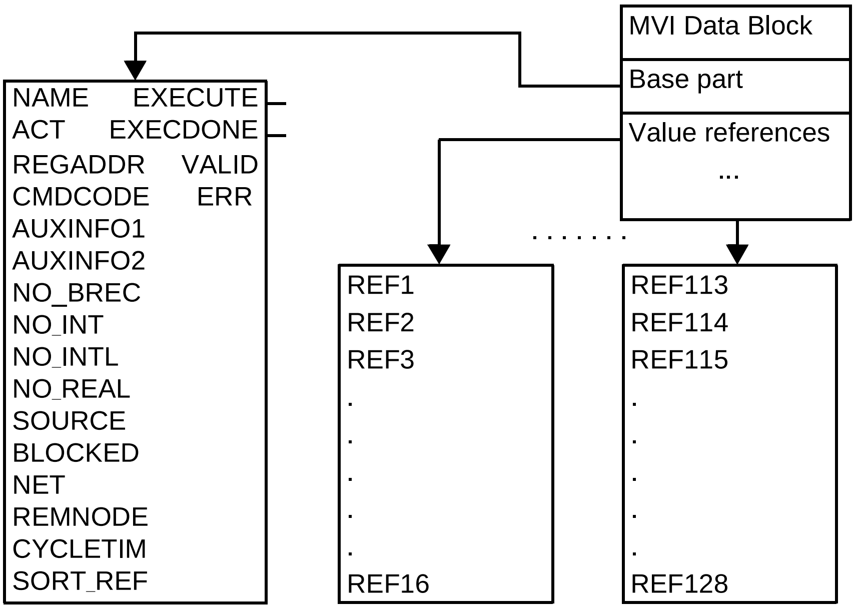


Figure 4-11. Representation of MVB Element, Overview

Special properties of the MVI Data Block DB element MVB (see Figure 4-11) are:

ACT	The ACT terminal specifies whether the element is active or not.
NET	Network number, it must be the same as in the corresponding MVINODE.

- REMNODE Remote node number, it must be the same as in the corresponding MVINODE.
  In master mode it is the number of the slave, 255 means global command.
  In slave mode it is number of the master.
  NET and REMNODE identify the MVB instance.
- CMDCODE, AUXINFO1, AUXINFO2 They define the code and auxiliary information of the command to be executed for RCOM.
- BLOCKED It defines whether the cyclic execution of the specified RCOM command is blocked.
- CYCLETIM The scan factor specifies the transmission interval of the MVB.

When you are creating an MVB element on an engineering station, you can specify the number of packed Boolean data DAT(B), integer data DAT(I), integer long data DAT(IL), and real data DAT(R) as NO_BREC, NO_INT, NO_INTL, and NO_REAL.

This causes the engineering station to automatically create the corresponding DAT elements. The NAME of the MVB element is used as the first part of the DAT name.

It is possible to make references to existing DAT elements when you are creating the MVB element on an engineering station. These references automatically declare the actual numbers (NO_BREC, …) of the DAT elements to which are referenced.

The numbers of DAT elements (NO_BREC, …) are justified if the number of references does not reflect the actual numbers.

For already existing MVB elements, you can delete existing references and add new references.

When you leave the MVB element on the engineering station, the references are stored in the order B, I, IL, R without any empty REFx, if SORT_REF = Yes.

NOTE

If the name of the MVB element is changed later on, the names of the corresponding DAT elements are not automatically updated!

The SOURCE terminal defines the direction of the communication (R=receive, S=send).

4.11.3 RCOM Network Configurations

RCOM is a protocol for communication between a superior station (master) and a subordinate station (slave).

The AC 110, 160, or derivatives can serve as master for another AC 110, 160, or 55 and as slave for MP200/1, Advant Controller 400 Series controllers, or another AC 110 or derivative.

In case of AC 110, 160, and derivatives you can connect a maximum of four telecommunication lines to the CI532 module and a further telecommunication line directly to the CPU module. The number of slave nodes for the whole AC 110, 160, and derivatives is limited to 50.

This section is not applicable for AC 70.

The RCOM interface in the AC 55 basic unit can be used as slave or as master. Possible partners are, for example, AC 110, AC 160, Advant Controller 400 Series, or MasterPiece 200/1. Since there is only one interface, the number of telecommunication lines is limited to one per Advant Controller 55. In master mode it supports up to 25 slave nodes.

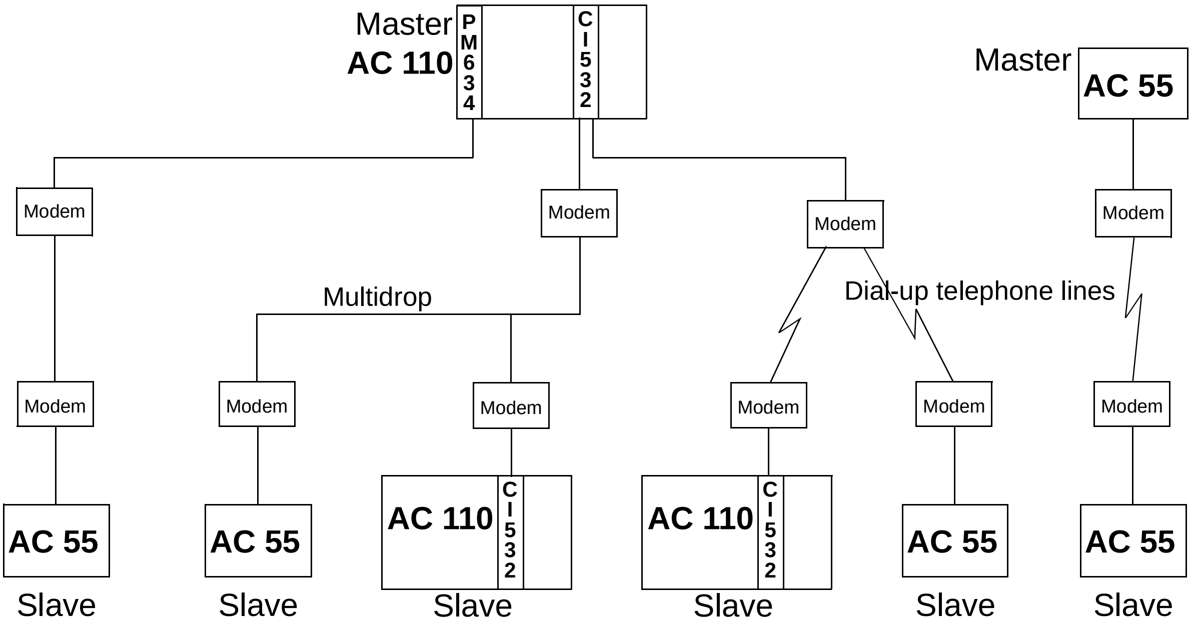


Figure 4-12. RCOM Network Configuration (Example)

You can build hierarchical network structures (see Figure 4-13).

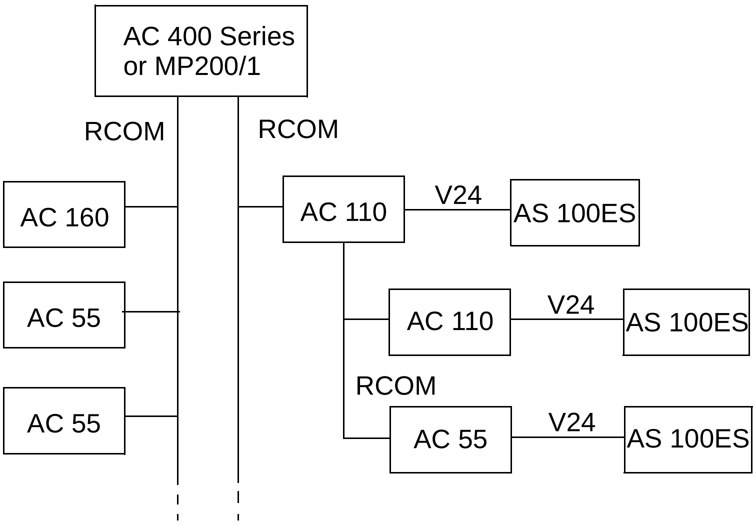


Figure 4-13. Hierarchical RCOM Network Configuration (Example)

AC 110, 160, 55, and derivatives cannot be programmed by an AS 100ES over an RCOM link. The engineering station must be connected directly by a V.24 line.

4.11.4 RCOM Functions

AC 110, 160, 55, and derivatives support the RCOM and RCOM+ protocols with the following functions:

- Data transmission

  Data is transmitted between the master and a slave in MVI Data Blocks (MVB). MVBs are special DataSets for the RCOM protocol. RCOM allows up to eight DATs per MVB and RCOM+ allows up to 24 DATs per MVB. The maximum number of MVBs is 340 for AC 55 and 500 for AC 110, 160, and derivatives.

- Control of slaves

  An AC 110, 160, 55, or derivative configured as master can control a slave by RCOM commands (cold start, warm start, and so on).

- Transfer of time-tagged events

  Time-tagged events can be transferred from an AC 110, 160, 55, or derivative configured as slave to an AC 110, 160, derivative, or Advant Controller 400 Series configured as master.
  Event transfer is specified by special DB elements. See Section 4.12, Event Handling.

- Time synchronization

  A RCOM channel in master mode distributes the actual time to all slaves, so the system time between the master and all slaves is synchronized.
  The protocols RCOM and RCOM+ differ only in the number of DATs per MVI Data Block (eight in RCOM and 24 in RCOM+) and in the break character handling.

4.12 Event Handling

AC 110, 160, and derivatives provide event detection and distribution of time-tagged events over AF 100 and RCOM.

AC 70 and derivatives provide event detection and distribution of time-tagged events over AF 100.

AC 55 provides event detection and distribution of time-tagged events over RCOM.

4.12.1 Overview

4.12.1.1 Event Detection

Kinds of Time-tagged Events

Advant Controller 100 Series controllers generate the following time-tagged events:

“Calculated” events which generation is configured by DB elements AIC and DIC.

In this section we describe only the common calculated events.

In case of AC 110, 160, and derivatives, there are events generated also by other sources:

- Events from I/O modules, such as DI650/DI651/DI652
- Events received in MVI Data Blocks and converted into the internal event format by means of DB elements AIMVB and DIMVB
- Events from extended DB elements, such as AIS, AOS.

DB elements DIC, AIC, DIS650/DIS651/DIS652, DIMVB, AIMVB used for configuration of event detection functions are called **Event Channel** elements.

“Calculated” Events

DB elements DIC and AIC allow to watch connected PC signals of type Boolean or Real for events. Event detection is “calculated” according to the parameters specified at these DB elements with a resolution down to 10 ms.

An event is detected from a Boolean PC signal when the signal changes from 0 to 1 or vice versa. A filter can be used to suppress event generation in case of signal spikes. If the filter is enabled the PC signal watched for events has to keep a new value for at least the *filter time* before the value change is accepted as an event.

From a PC signal of type Real an event is detected when the signal exceeds one of the two low or high limits or moves back into the range (see Figure 4-14). The hysteresis is used to suppress event generation when the signal just fluctuates around a limit.

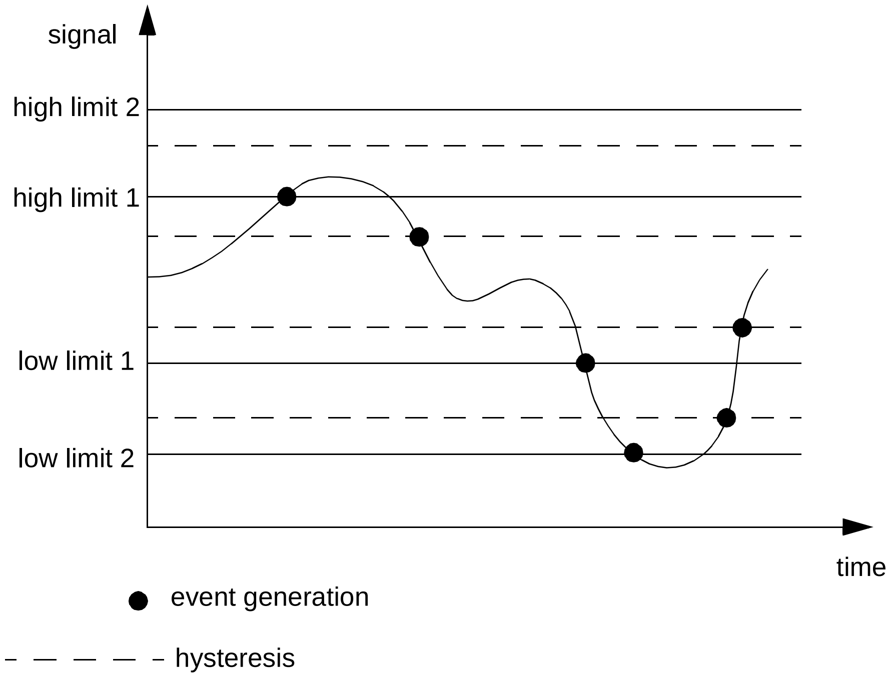


Figure 4-14. Event Detection for PC Signals of Type Real

Event Queues

Events detected on controller are buffered in up to 32 event queues. Each event queue can store up to 500 events. If events are generated while the event queue is full the new events overwrite the oldest events. The event queue is represented by the EVS DB element.

4.12.1.2 Event Transfer

General Features

Depending on the type of controller, time-tagged events from a controller are transferred to one or several event receivers over Advant Fieldbus 100 or an RCOM link. An event receiver can be:

- AC 110, 160, and derivatives
- Advant Controller 400 Series
- A personal computer running a Human-Machine Interface system based on, for example, AdvaSoft for Windows or Advant OPC Server.

NOTE

The number of event receivers is limited to 10 for AF 100 and to 1 for RCOM.

A possible network configuration for event transfer is shown in Figure 4-15:

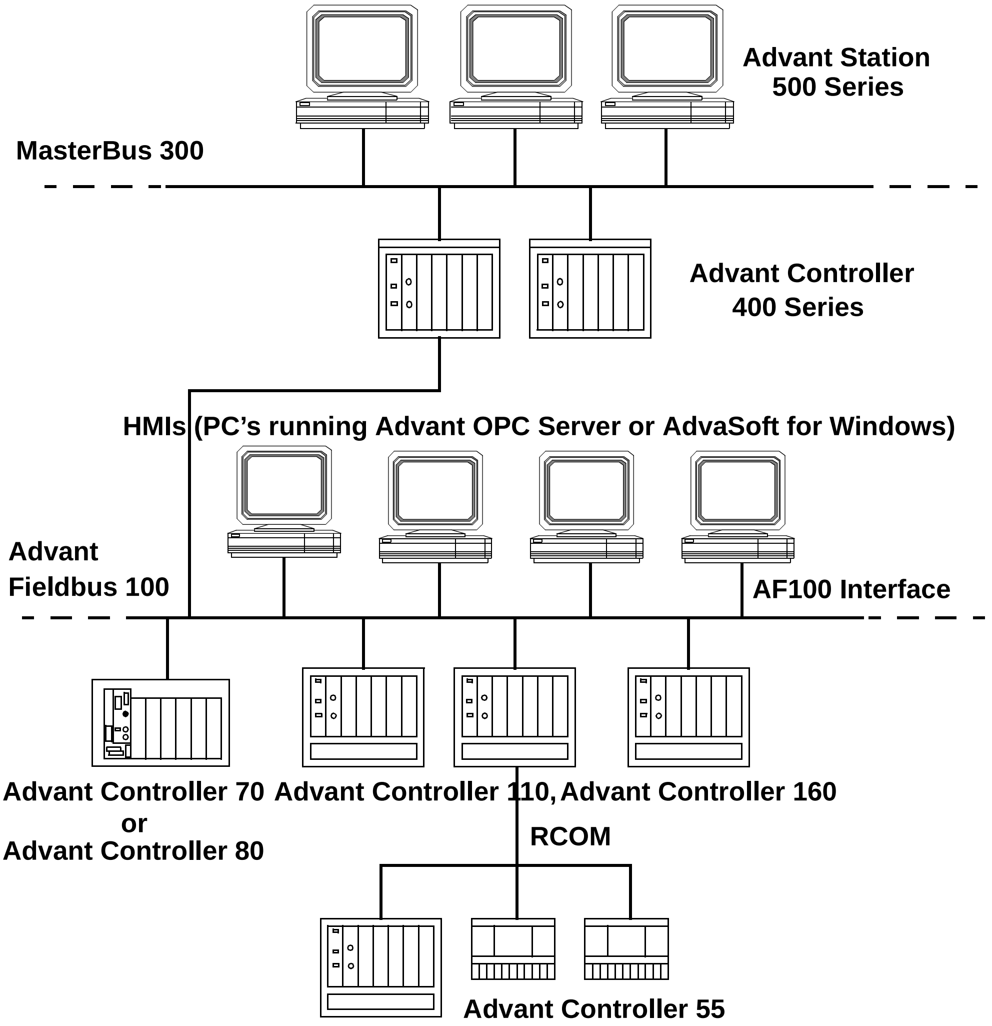


Figure 4-15. Configuration Alternatives for Event Transfer

The transfer of time-tagged events is configured by Event Set (EVS) elements.

An Event Set element of type “send” (configured by data base element EVS(S)) groups a set of Event Channels (AIC, DIC, and so on, as applicable) for transmission of events. Each Event Set can handle up to 32 Event Channels of different types which can be mixed in arbitrary order.

An EVS(S) element collects events from its referenced Event Channel elements and sends the events to the event receivers when those requests them.

An Event Set of type “transit” (configured by data base EVS(T)) is used in AC 110 and 160 for transiting (receiving and sending) events from an EVS(S) element located on a remote RCOM node to an event receiver in another RCOM network or to an event receiver at the Advant Fieldbus 100. The AC 110, 160, or derivative then acts as a transit station.
The EVS(T) element has to refer to the opposite sending EVS(S) element. This is configured by specifying the RCOM network address net/node of the remote node and the identity of EVS(S).

An AC 110, 160, or derivative can handle up to 32 Event Set elements EVS(S) or EVS(T). For DB elements EVS(S) the Event Channel elements (AIC, DIC, DIS650/DIS651/DIS652, AIMVB, DIMVB as applicable) are considered as event sources. For DB elements EVS(T) the referenced DB element EVS(S) is considered as event source.

An AC 70, AC 55, or derivative can handle up to 32 Event Set elements EVS(S).

Event Transfer to HMI

AC 110, 160, 70, and derivatives transfer events directly to a personal computer running Advant OPC Server or AdvaSoft for Windows via AF 100.

AC 110, 160, 55, and derivatives transfer events directly to a personal computer running Advant OPC Server or AdvaSoft for Windows via RCOM+.

Events generated in an AC 55 can be transferred to a personal computer running Advant OPC Server or AdvaSoft for Windows also via an AC 110, 160, or derivative as transiting station (see Figure 4-15).

Event Transfer to Advant Controller 400 Series

Event Set elements of the “receive” type (configured by data base element EVS(R)) are used in the Advant Controller 400 Series to receive events. The EVS(R) element must point to the opposite sending EVS(S) or EVS(T) element in the controller.

See Figure 4-16 for an overview of the EVS data base element relationship.

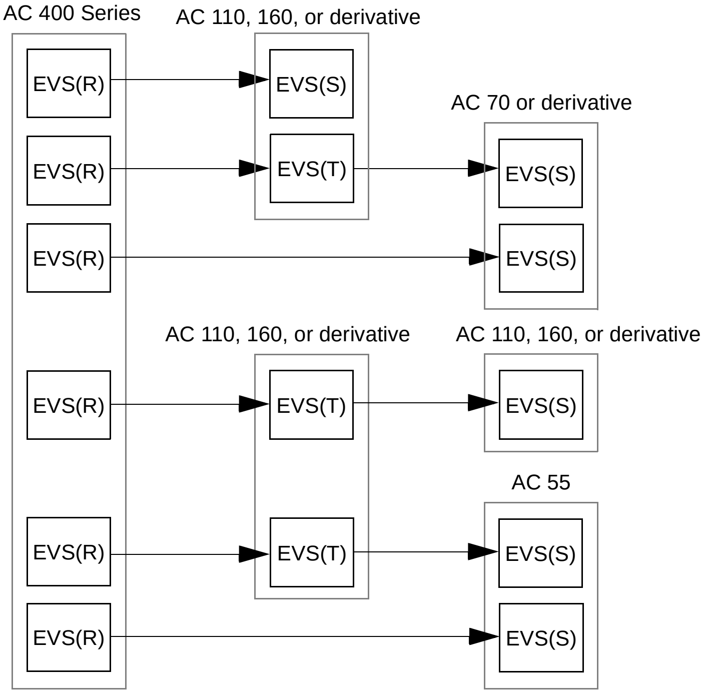


Figure 4-16. Event Set (EVS) Element Relationship

An EVS(S) element on the controller contains references (terminals REF1 to REF32) to Event Channel elements AIC, DIC, and if applicable, DIS650/DIS651/DIS652, AIMVB, or DIMVB. An EVS(R) element on the Advant Controller 400 Series contains references (terminals REF1 to REF32) to Event Channel elements AIEV or DIEV.

See Figure 4-17 and Figure 4-18 for an overview of the relationship between Event Channels in the Advant Controller 400 Series and Advant Controller 100 Series.

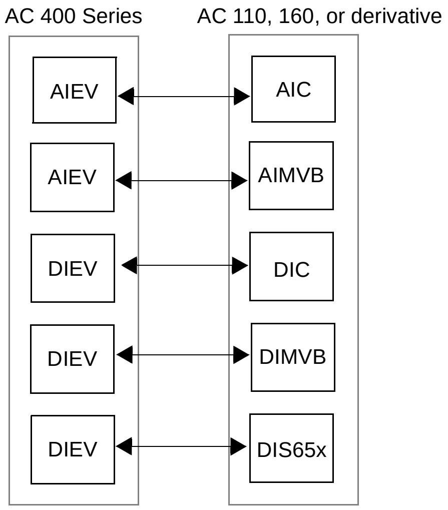


Figure 4-17. Event Channel Relationship Advant Controller 400 Series and AC 110 or 160

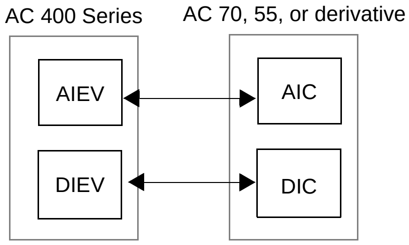


Figure 4-18. Event Channel Relationship AC 400 Series and AC 70 or AC 55

It is important to specify matching Event Channel elements on the corresponding reference terminal (REF1 to REF32) on both sides (Controller and Advant Controller 400 Series) in the Event Set elements EVS(S) and EVS(R).

EVS(T) element does not contain any references to Event Channel elements because it only transits the events further.

Configuration Examples

In Figure 4-19, Figure 4-20, and Figure 4-21 three configuration examples are shown for event transfer from Advant Controller 100 Series controller to an Advant Controller 400 Series controller or a PC running AdvaSoft for Windows, using AF 100 or RCOM or both.

In Figure 4-21 an example of a network configuration is shown for event transfer over AF 100 from AC 110 and 70 to Advant Controller 400 Series station:

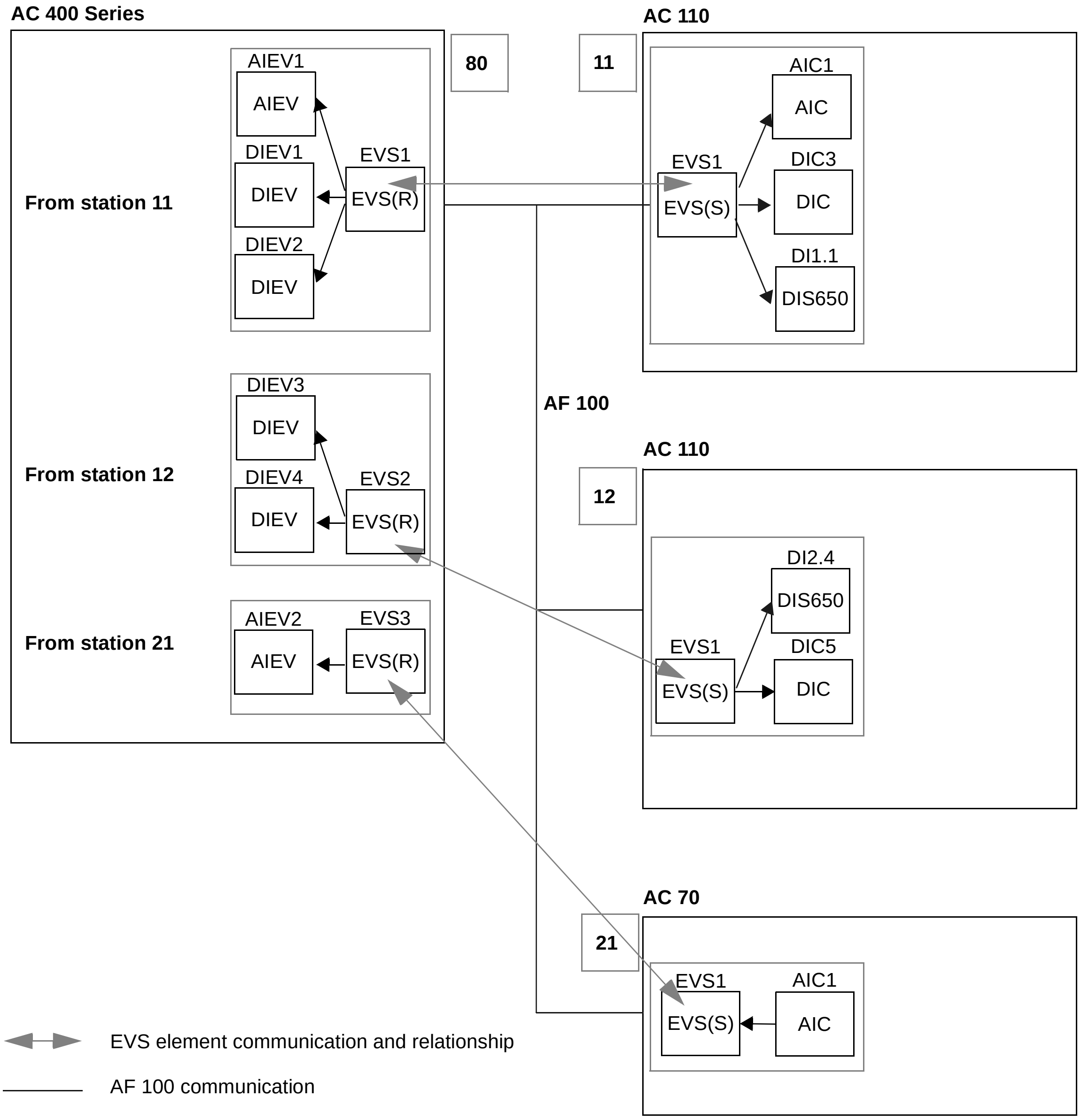


Figure 4-19. Event Transfer via AF 100 from AC 110 and 70 to AC 400 Series (Example)

In Figure 4-20 an example of a network configuration is shown for event transfer over RCOM. Events from AC 55 with RCOM node address 11 and AC 110 with RCOM node address 12 are directly transmitted to the Advant Controller 400 Series station.
Events from AC 55 with RCOM node address 21 are transmitted over the AC 110 to the Advant Controller 400 Series station.

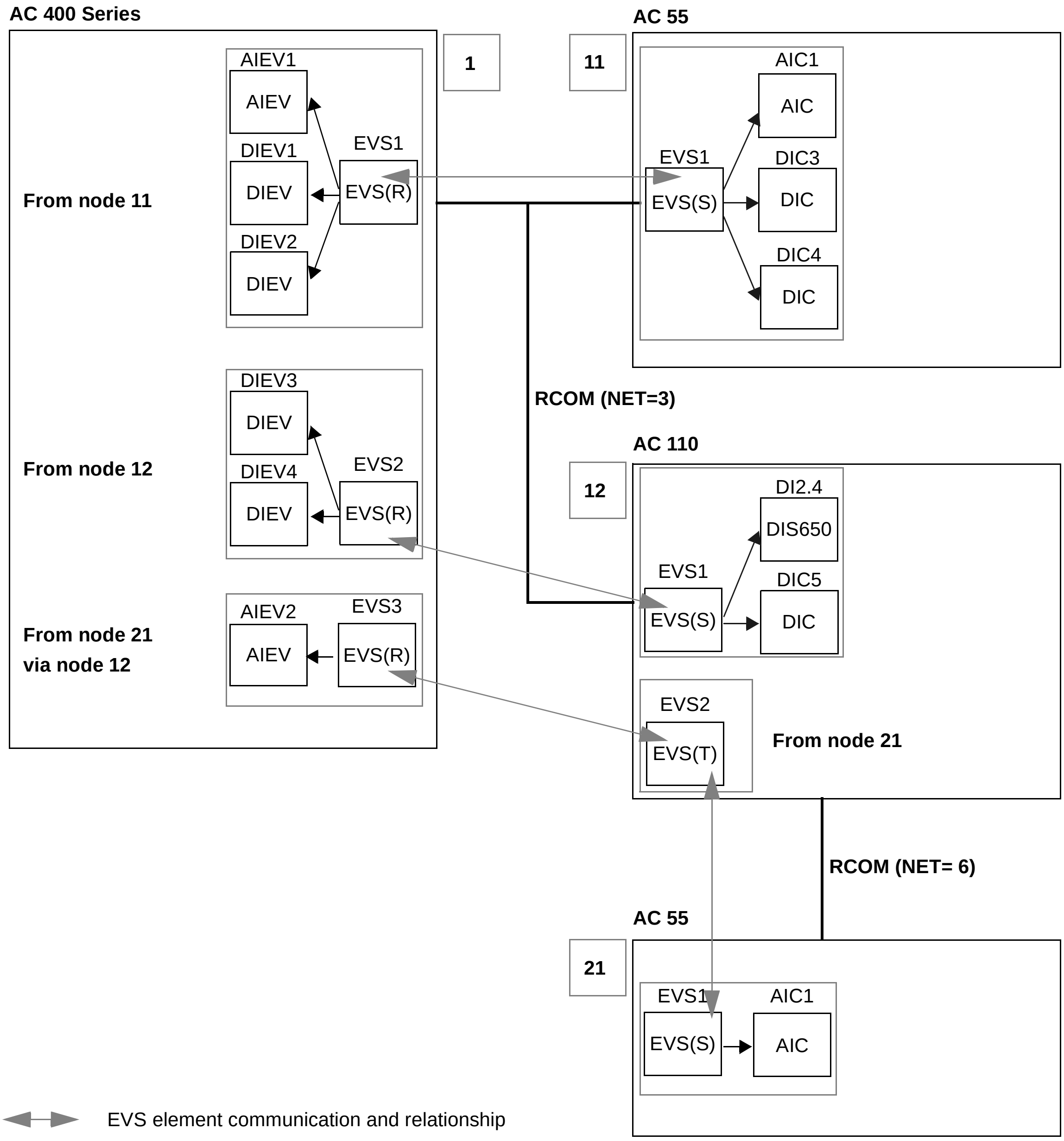


Figure 4-20. Event Transfer via RCOM from AC 110 and 55 to AC 400 Series (Example)

In Figure 4-21 an example of a network configuration is shown for event transfer to a PC running AdvaSoft for Windows using both AF 100 and RCOM:

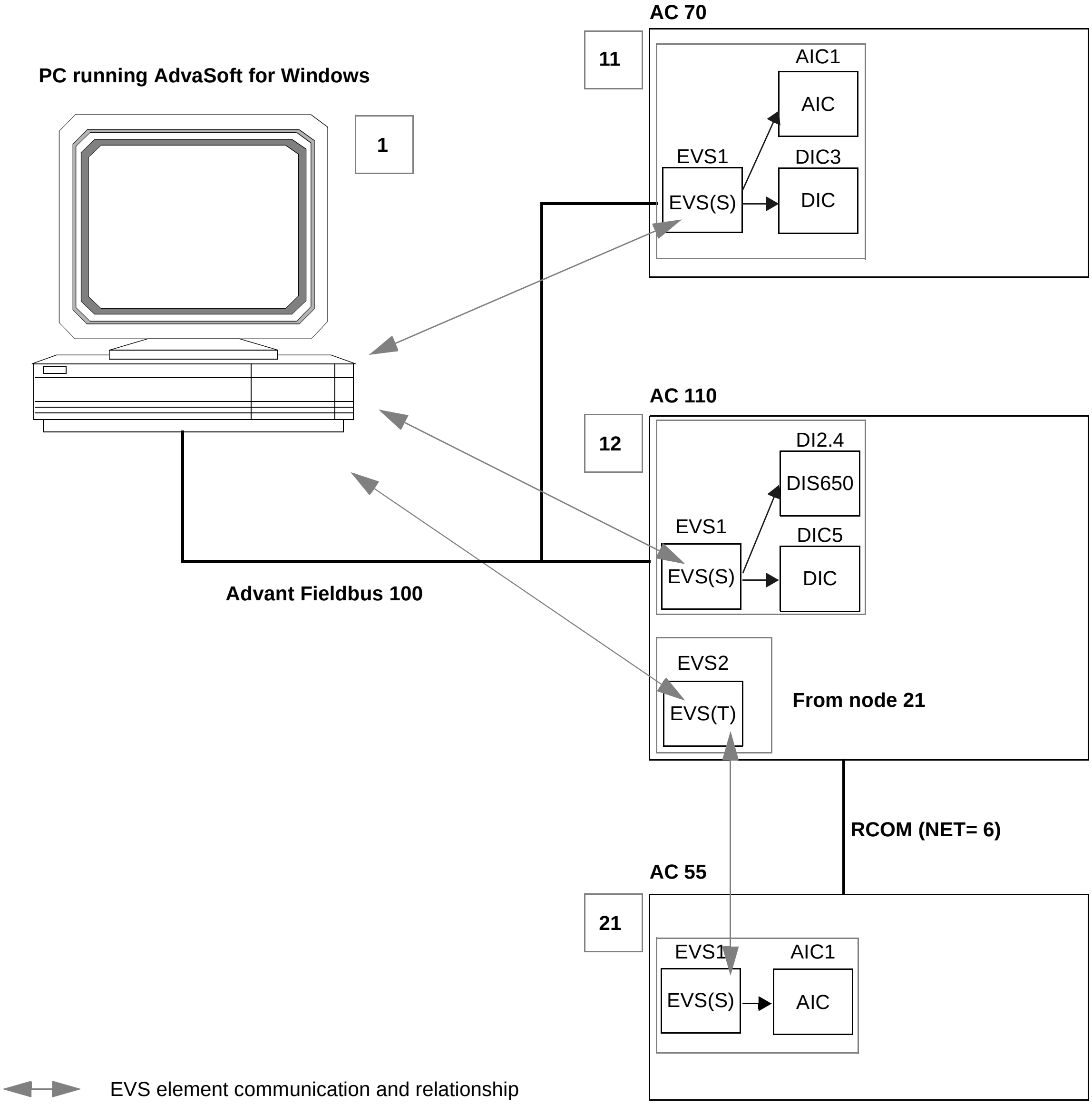


Figure 4-21. Event Transfer from AC 110, 70, and 55 to PC with AdvaSoft for Windows (Example)

4.12.2 Configuration of Event Sets

This section gives a more detailed description on how to configure Event Sets.
We describe the following procedures:

- Create and configure the Event Channel elements AIC, DIC.
- Create and configure the Event Set elements EVS(S) and for AC 110 and 160 also EVS(T).

For a description of how to configure the AIMVB, DIMVB, and DIS650/DIS651/DIS652 elements, see *Advant Controller 110 User's Guide*.

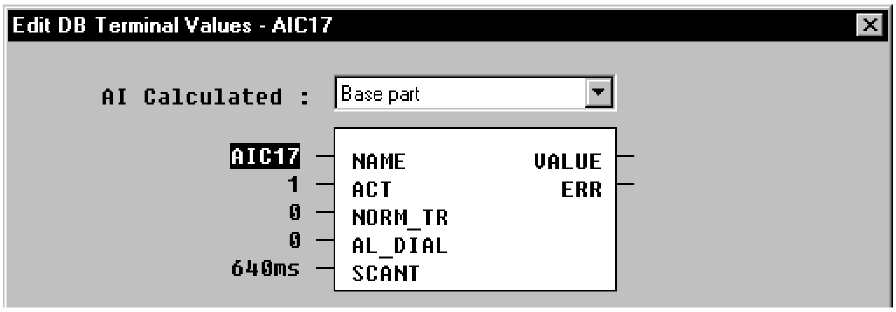
For a description of how to configure the Event Set element EVS(R), see *AMPL Configuration Advant Controller 400 Series Reference Manual*.

4.12.2.1 Configuration of Event Channel Elements

Event Channel Element AIC

The AIC element is used to detect analog type calculated events. An event is detected when the value of the terminal VALUE exceeds one of the two high or low limits or moves back into the range.

Event detection is enabled as long as the terminal NORM_TR is set to 1. The period by which the terminal VALUE is cyclically scanned for events (scan time) is specified by terminal SCANT. It is recommended that you use as big scan time as possible from the application point of view because the bigger the scan time, the lower the system load.



Edit DB Terminal Values - AIC17

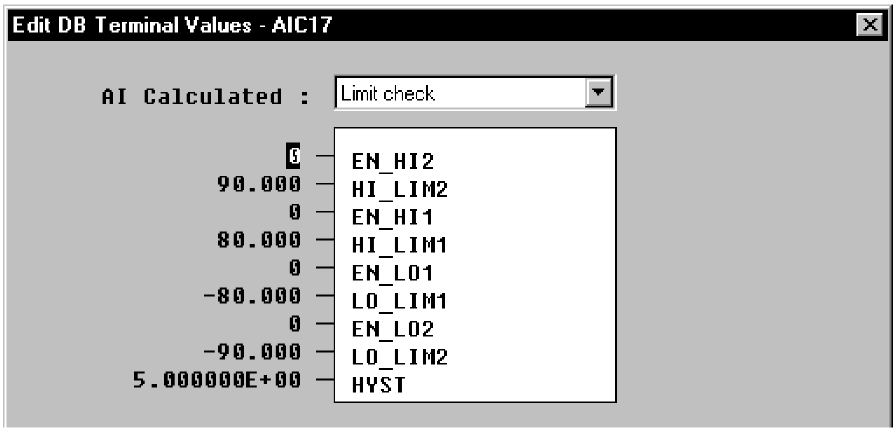
AI Calculated : Base part

AIC17 — NAME
1 — ACT
0 — NORM_TR
0 — AL_DIAL
640ms — SCANT
VALUE —
ERR —

Figure 4-22. AIC Element, Base Part (Engineering Station View)

If events from an Event Channel element are transferred over a remote communication link (RCOM) with a dial-up line, the terminal AL_DIAL enables the alarm dial feature. This feature allows the controller to dial up the master node every time an event is detected by this Event Channel.

For more information about RCOM, including alarm dialing, see *RCOM Advant Controller 110 User´s Guide* or *RCOM Advant Controller 55 User´s Guide*.



Edit DB Terminal Values - AIC17

AI Calculated : Limit check

0	EN_HI2
90.000	HI_LIM2
0	EN_HI1
80.000	HI_LIM1
0	EN_LO1
-80.000	LO_LIM1
0	EN_LO2
-90.000	LO_LIM2
5.000000E+00	HYST

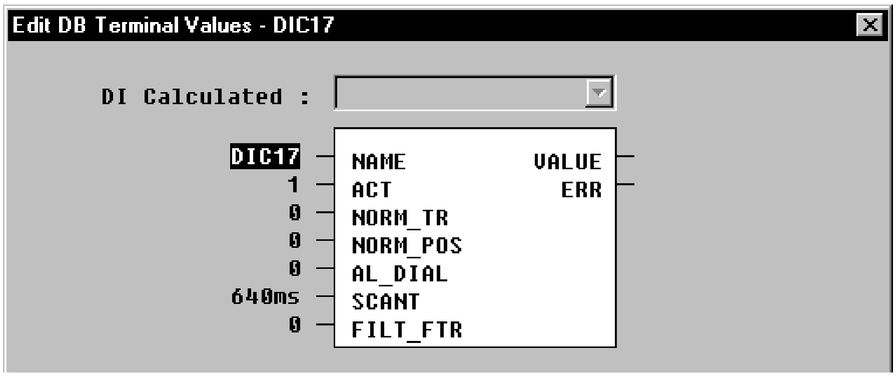
Figure 4-23. AIC Element, Limit Check (Engineering Station View)

Each limit can be enabled or disabled independently of other limits. Check that the condition LO_LIM2 < LO_LIM1 < HI_LIM1 < HI_LIM2 is fulfilled. Otherwise, the sequence of events caused by limit crossing is undefined. You can use the hysteresis parameter (terminal HYST) to suppress event generation if the value is fluctuating around a limit.

Terminals NORM_TR, SCANT, and AL_DIAL are used for Event Channel elements DIC, AIMVB, and DIMVB in the same way.

Event Channel Element DIC

The DIC element is an Event Channel element used for detection of digital type calculated events.



Edit DB Terminal Values - DIC17

DI Calculated :

DIC17	NAME	VALUE
1	ACT	ERR
0	NORM_TR	
0	NORM_POS	
0	AL_DIAL	
640ms	SCANT	
0	FILT_FTR	

Figure 4-24. DIC Element (Engineering Station View)

An event is detected when the value of terminal VALUE (Boolean type) changes from 0 to 1 or vice versa and event generation is enabled (terminal NORM_TR=1).

The terminal NORM_POS specifies the initial value of terminal VALUE which is expected after deblocking of event detection. If the initial value is different from NORM_POS, an event is generated once.

A filter facility is used to suppress event generation in case of rapidly changing values.
The terminal VALUE must keep a new value for at least the filter time before the value change is accepted as an event. The filter time is calculated by the following equation:

filter time = FILT_FTR * SCANT.

If terminal FILT_FTR equals 0, the filter facility is disabled.

Event Channel Elements DIS650/DIS651/DIS652

This element is available only for AC 110, 160, and derivatives.

Before an element DIS650/DI651/DI652 can be configured, the data base element DI650/DI651/DI652 must be created and configured.

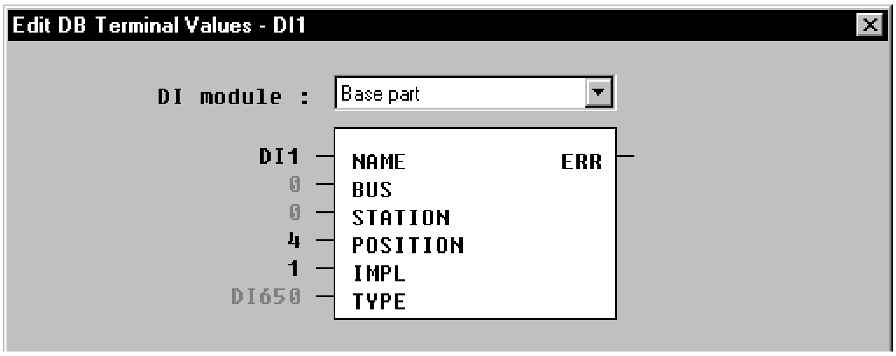
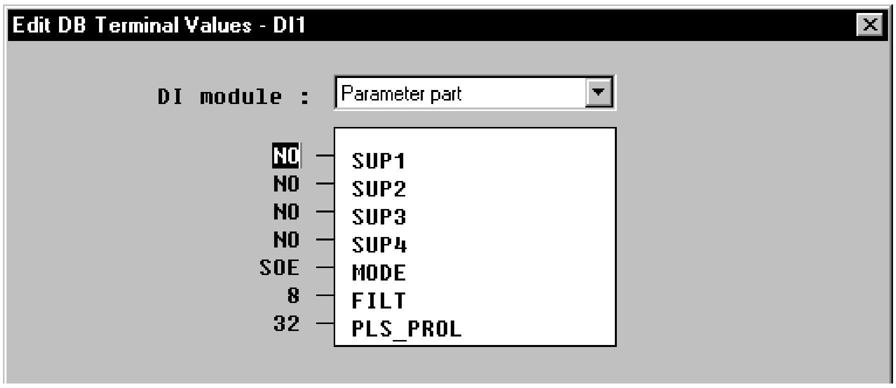


Figure 4-25. DI650/DI651/DI652 Module, Base Part (Engineering Station View)

The I/O module DI650/DI651/DI652 includes an SOE facility. The “sequence of event” (SOE) facility provides event detection and recording for every channel with a resolution of 1 ms. It is enabled by setting the module in mode SOE (terminal MODE).



DI module : Parameter part

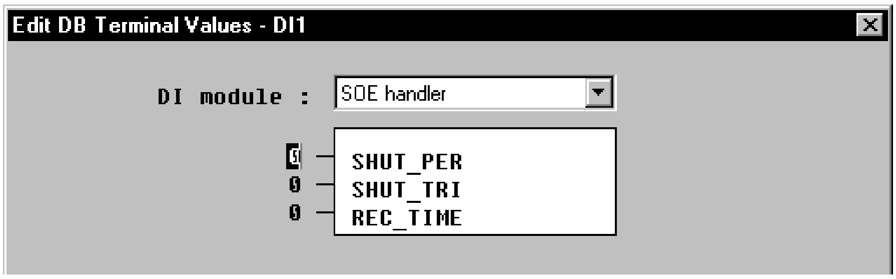
NO	SUP1
NO	SUP2
NO	SUP3
NO	SUP4
SOE	MODE
8	FILT
32	PLS_PROL

Figure 4-26. DI650/DI651/DI652 Module, Parameter Part (Engineering Station View)

The DI650/DI651/DI652 stores events in an internal event queue, the size of which is 64 entries. This event queue can be cleared with the terminal CLEAR_Q at the data base element EVS(S) if the EVS(S) references to at least one channel element of this module.

To avoid an overflow of the internal event queue due to a fast-changing process signal connected to a DI650/DI651/DI652 channel, the SOE facility includes a shutter filter.

The shutter filter is closed as soon as the number of events generated within the shutter period (terminal SHUT_PER) exceeds the shutter trigger number (terminal SHUT_TRI). In this case, the shutter filter does not pass any event as long as the recovery time (terminal REC_TIME) has not been expired. You can disable the shutter filter by setting SHUT_TRI to 0.

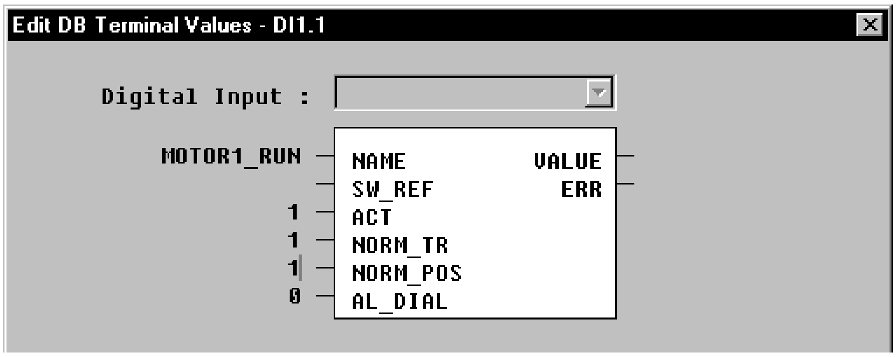


DI module : SOE handler

0	SHUT_PER
0	SHUT_TRI
0	REC_TIME

Figure 4-27. DI650/DI651/DI652 Module, SOE Handler (Engineering Station View)

When you are creating a module element DI650/DI651/DI652, 32 subordinated channel elements DIS650/DIS651/DIS652 are automatically created by the engineering station (see Figure 4-28).



MOTOR1_RUN — NAME

1 — ACT

1 — NORM_TR

1 — NORM_POS

0 — AL_DIAL

Figure 4-28. DIS650/DIS651/DIS652 Channel Element (Engineering Station View)

The terminal NORM_TR enables or disables event detection for this channel.
The terminal NORM_POS specifies the initial value of terminal VALUE which is expected after configuration of the I/O module.
If the initial value is different from NORM_POS, an event is generated once.

It is expected by the tool that every element DIS650/DI651/DI652 with terminal NORM_TR=1 is referenced by an EVS(S) element.

4.12.2.2 Configuration of Event Set Elements

The controllers support the Event Set element EVS(S).

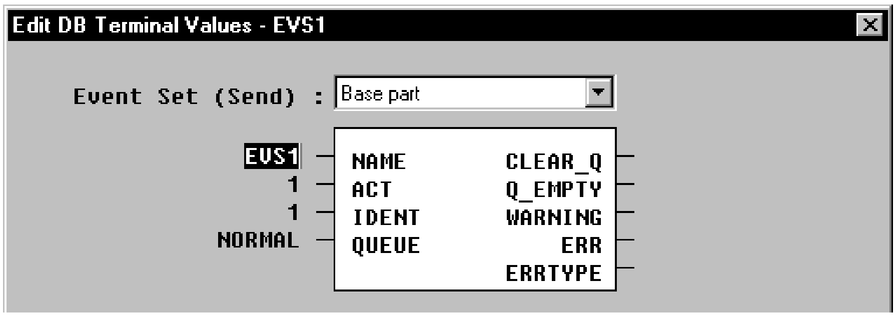
AC 110, 160, and derivatives support also the Event Set element EVS(T).
The maximum number of Event Set elements is 32 per controller, independently of the type of Event Set element.

Event Set Element EVS(S)

An EVS(S) element collects events from up to 32 referenced Event Channels in its internal event queue.

In case of AC 110, 160,, 70, and derivatives the Event Set sends the events to one or several event receivers (up to 10) when the event receivers requests the events. All event receiver are handled independently from each other.

In case of AC 55 the Event Set sends the events to an event receiver when the event receiver requests the events.


Edit DB Terminal Values - EVS1

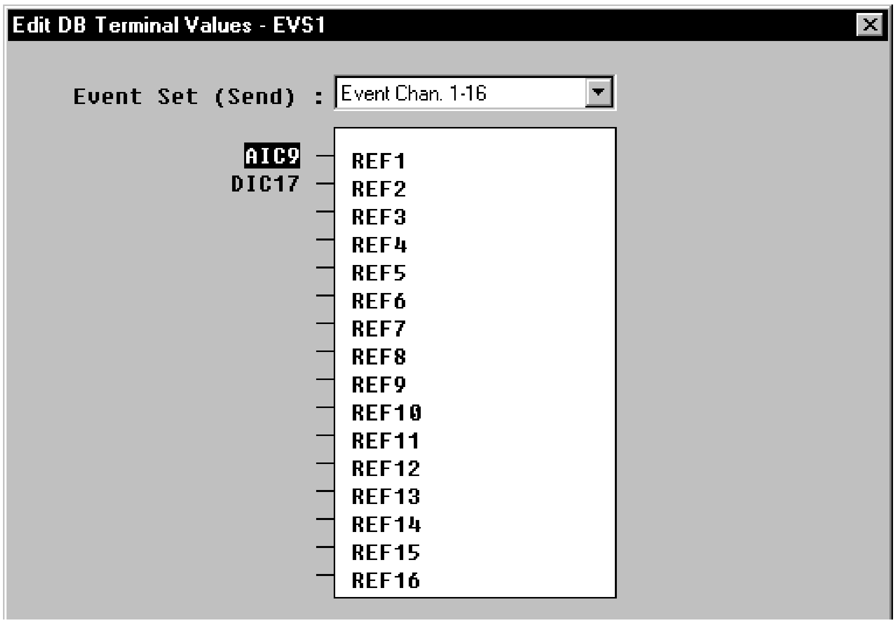
Event Set (Send) : Base part

EVS1	NAME	CLEAR_Q	
1	ACT	Q_EMPTY	
1	IDENT	WARNING	
NORMAL	QUEUE	ERR	
		ERRTYPE	

Figure 4-29. EVS(S) Element, Base Part (Engineering Station View)

Because the number of Event Set elements is limited to 32 per controller, the maximum number of referenced Event Channels is 1024, that is a maximum of 1024 signals can be configured for event detection.

The reference terminals (REF1 to REF32) are filled in with references to Event Channel elements of type AIC or DIC, in case of AC 110 and 160 also DIS650/DIS651/DIS652, AIMVB, or DIMVB. The name of the Event Channel element is specified as the reference, see the example in Figure 4-30.


Edit DB Terminal Values - EVS1

Event Set (Send) : Event Chan. 1-16

AIC9	REF1
DIC17	REF2
	REF3
	REF4
	REF5
	REF6
	REF7
	REF8
	REF9
	REF10
	REF11
	REF12
	REF13
	REF14
	REF15
	REF16

Figure 4-30. EVS(S), Event Channel References 1-16 (Example)

The relationship between the EVS(S) and the Event channel elements is shown in Figure 4-31.

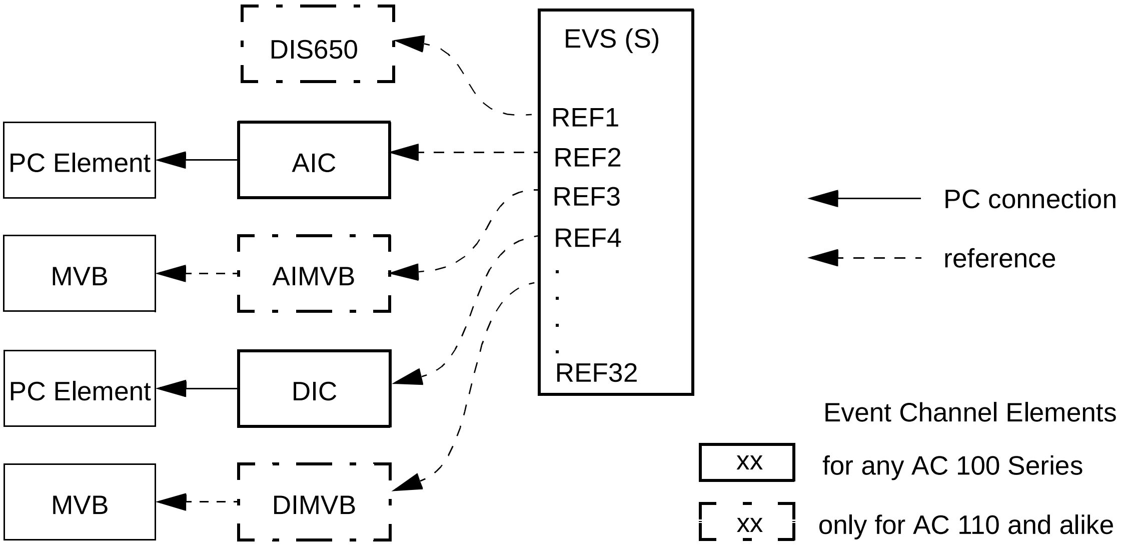


Figure 4-31. Relation Between EVS(S) and Event Channel Elements (Example)

An event queue is assigned to each EVS(S) element. The event queue is organized in FIFO (first in - first out) order. The event queue of the EVS(S) element stores all incoming events detected by the referenced Event Channel elements.

Specify the event queue size by terminal QUEUE. The value NORMAL means that the event queue's size is 100 event entries and EXTENDED means the event queue's size is 500 entries. An extended event queue is recommended only if you are transferring events over a RCOM link with a dial-up line. Else choose QUEUE=NORMAL. In this way, less memory is consumed for the event queue and more memory is available for application programming.

Event queues are normally protected against power failure. The duration of the protection depends on the type of controller.

The terminal Q_EMPTY of an Event Set is 1 when the event queue is empty.

You can clear an event queue of an Event Set by setting the terminal CLEAR_Q **until** the terminal Q_EMPTY becomes 1.

In case of AC 110, 160, and derivatives, if more than one event receiver is requesting events from an EVS the terminal Q_EMPTY gets 1 only when **all** event receivers have read all events from the event queue. If Event Channel elements of type DIS650/DIS651/DIS652 are referenced, the internal event queue of the corresponding DI650/DI651/DI652 module is also cleared.

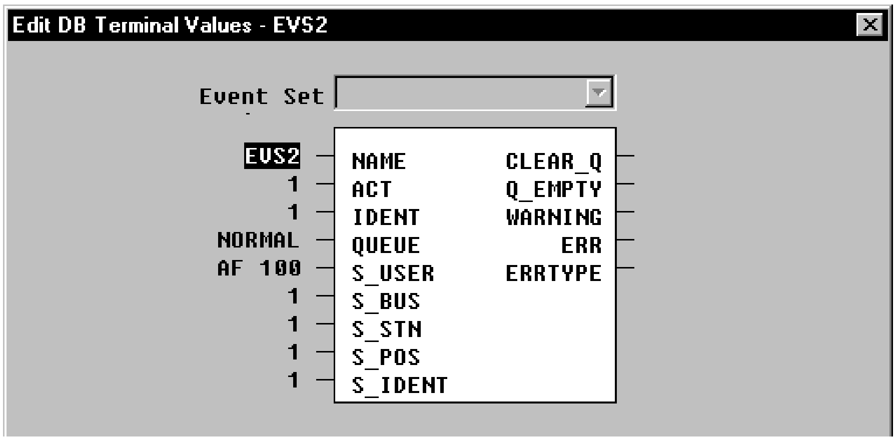
NOTE

Channel elements of one DI650/DI651/DI652 I/O module can be referenced only by one EVS(S) element.

Event Set Element EVS(T)

This element is available only for AC 110, 160, and derivatives.
An Event Set of type “transit” (configured by data base element EVS(T)) is used to transit events from an EVS(S) element located on a remote RCOM node to an event receiver in another RCOM network (for a configuration with a hierarchical RCOM network see Figure 4-20) or to one or several event receivers (up to 10) at Advant Fieldbus 100.

EVS(T) has to refer to the opposite sending EVS(S) element. Because the number of Event Set elements is limited to 32, an AC 110, 160, or derivative can transit events from up to 32 other controllers.



Edit DB Terminal Values - EVS2

Event Set

EVS2	NAME	CLEAR_Q	
1	ACT	Q_EMPTY	
1	IDENT	WARNING	
NORMAL	QUEUE	ERR	
AF 100	S_USER	ERRTYPE	
1	S_BUS		
1	S_STN		
1	S_POS		
1	S_IDENT		

Figure 4-32. EVS(T) Element (Engineering Station View)

Terminals S_USER are filled in with the used communication type AF 100 (=Advant Fieldbus 100) or RCOM (=RCOM or RCOM Extended). S_IDENT specifies the identity of the EVS(S) from which events are requested and then transited further to the EVS(R) element on the Advant Controller 400 Series. Terminals S_BUS and S_STN specify the address bus/station for AF 100 (net/node for RCOM or RCOM Extended) where this EVS(S) element is located.
Note that S_USER is normally RCOM because there is no need to transit events from a station connected to AF 100 (except for test purposes).

An event queue is assigned to each EVS(T) element. The event queue is organized in FIFO (first in - first out) order. All incoming events received from the referenced EVS(S) are stored in the event queue of the EVS(T) element before they are transited further.

You can specify the size of the event queue by terminal QUEUE. The value NORMAL means that the event queue’s size is 100 event entries and EXTENDED means the event queue’s size is 500 entries.

An extended event queue is recommended only if you are transferring events over an RCOM link with a dial-up line. In all other cases, choose QUEUE=NORMAL. In this way, less memory is consumed for the event queue and more memory is available for application programming.

If the battery is plugged in, event queues are protected against power failure.

The terminal Q_EMPTY of an Event Set is 1 when the event queue is empty. If more than one event receiver is requesting events from an EVS the terminal Q_EMPTY gets 1 only when **all** event receivers have read all events from the event queue.

You can clear an event queue of Event Set by setting the terminal CLEAR_Q **until** the terminal Q_EMPTY becomes 1.

If events are transmitted over an RCOM link, take into account that the transmission of events has a lower priority than the transmission of MVI Data Blocks. How to build the application in this case is described in *RCOM Advant Controller 110 User's Guide*.

Diagnostics of Event Sets

Diagnostics information for Event Sets is available at the terminals ERR, WARNING, and ERRTYPE. All three terminals indicate errors in the transmission of events **to** the event receivers, and in case of AC 110 and 160 also **from** event sources.

In case of AC 110, 160, and derivatives, the Event Channel elements DIS650/DIS651/DIS652 are considered as event sources for DB elements EVS(S). For DB elements EVS(T) the referenced DB element EVS(S) is considered as event source.

The terminal ERR indicates that events can not be transferred to an event receiver, or in case of AC 110, 160, and derivatives also can not be received from an event source.
In addition, a full event queue is considered as an error.

The terminal WARNING indicates that the transmission of events to an event receiver is disturbed.

The terminal ERRTYPE is of type integer and thus includes 16 bits. These bits give a more detailed error indication (see Table 4-6) than the terminals ERR and WARNING.

Table 4-6. Bits in EVS Terminal ERRTYPE

Bit	Error Description	EVS Terminals ERR, WARNING
0	Only in case of AC 110, 160, and derivatives: Communication to an event source is disturbed	WARNING = 1
1	Communication to an event receiver is disturbed	WARNING = 1
2	Only in case of AC 110, 160, and derivatives: Event source error	ERR = 1
3	Event receiver error	ERR = 1
4	Event queue filled for at least 80% (for at least one event receiver)	WARNING = 1
5	Event queue overflow (for at least one event receiver)	ERR = 1
6 to 15	Not used	

4.12.2.3 Event Transfer via Hierarchical Network (Example)

In Figure 4-20 an example of event transfer over a hierarchical RCOM network is shown. The configuration of the Event Sets for this example is described below.

Node 21

In node 21, the EVS(S) element EVS1 collects events from Event Channel element AIC1.

Table 4-7. Terminals of EVS(S) Element EVS1

Terminals	Value	Comment
IDENT	7	

Node 12

In node 12, the EVS(S) element EVS1 collects events from local Event Channel element DIC5 and DI2.4.

Table 4-8. Terminals of EVS(S) Element EVS1

Terminals	Value	Comment
IDENT	2	

EVS(T) element EVS2 transits events received from EVS(S) with IDENT=7 on node 21 in network 6.

Table 4-9. Terminals of EVS(T) Element EVS2

Terminals	Value	Comment
IDENT	3	
S_USER	RCOM	
S_BUS	6	RCOM network number
S_STN	21	RCOM node number
S_IDENT	7	

Node 11

In node 11, the EVS(S) element EVS1 collects events from local Event Channel elements AIC1, DIC3, and DIC4.

Table 4-10. Terminals of EVS(S) Element EVS1

Terminals	Value	Comment
IDENT	2	

Node 1

EVS(R) element EVS1 receives events from EVS(S) with IDENT=2 on node 11 in network 3.

Table 4-11. Terminals of EVS(R) Element EVS1

Terminals	Value	Comment
IDENT	1	
S_USER	RCOM	
S_BUS	3	RCOM network number
S_STN	11	RCOM node number
S_IDENT	2	

EVS(R) element EVS2 receives events from EVS(T) with IDENT=2 on node 12 in network 3.

Table 4-12. Terminals of EVS(R) Element EVS2

Terminals	Value	Comment
IDENT	2	
S_USER	RCOM	
S_BUS	3	RCOM network number
S_STN	12	RCOM node number
S_IDENT	2	

EVS(R) element EVS3 receives events from EVS(T) with IDENT=3 on node 12 in network 3.

Table 4-13. Terminals of EVS(R) element EVS3

Terminals	Value	Comment
IDENT	3	
S_USER	RCOM	
S_BUS	3	RCOM network number
S_STN	12	RCOM node number
S_IDENT	3	

4.13 Configuration of Extended DB Elements

4.13.1 Overview

Extended DB elements are used to simplify the communication between the controller and AdvaSoft for Window as well as the communication between different controllers.

The extended DB element is a concept that facilitates the engineering of AC 110, 160, and derivatives.

It addresses the following issues:

- Definition of signals rather independently of the I/O hardware.
- Definition of signals and texts for HMI (MMI) purposes. Every aspect of the signal, from DSP communication up to event generation can be seen at a single DB element.
- Distribution of signals on the AF 100. There is only one point in the system where the signal is defined. The definition includes also the conversion of raw values to engineering units.

The intention behind extended DB elements is the single point data entry in order to configure the controller as well as the HMI.

Extended DB elements carry information concerning the following items:

- Controller configuration
- HMI (MMI) configuration
- Communication
- Signal description.

The following extended DB elements are available for AC 110 and 160:

- Input elements
  - Analog input (AIS)
  - Digital input (DIS.
- Output elements
  - Analog output (AOS)
  - Digital output (DOS).
- Memory elements (MB, MI, MIL, MR).

To work with extended DB elements with all their advantages, you need the following three systems:

- Controller with base software or options supporting extended DB elements, such as AC 110 (version 2.1 or higher) or AC 160
- AdvaSoft for Windows (version 1.3) or other newer HMI system based on Advant OPC Server
- Engineering environment supporting extended DB elements, such as Function Chart Builder version 4.3/0 or higher.

If you create, modify, and delete extended DB elements, FCB automatically creates, modifies, and deletes other DB elements, as required by the extended DB element.

Configuration of extended DB elements is done in two ways, depending on whether the configuration parameter is changeable at run-time by HMI:

- Some terminals have fixed values that can only be changed by FCB in on-line mode.
- Some terminal values can be changed by HMI.
  For example, HMI can initialize them at run-time by writing initial values.

We describe the usage of extended DB elements from the following points of view:

- Selection of signal sources, connection to signal sources
  (see Section 4.13.3.2, Selecting HW (Channel Element) as Source
  to Section 4.13.3.7, Engineering Station as Source)
- Internal processing in extended DB elements
  (see Section 4.13.4, Processing in Extended DB Elements)
- Connection to signal sinks
  (see Section 4.13.5, Connection to Signal Sinks).

Reference information on extended DB elements is summarized in Section 4.13.7, Special Terminals of Extended DB Elements.

Section 4.13.8, Example of Extended DB Element Configuration is added for illustration of extended DB element usage.

The following figures shows the main system components participating in the realization of extended DB element concept. One of the examples contains Advant OPC Server-based HMI, the other one contains AdvaSoft for Windows as HMI.

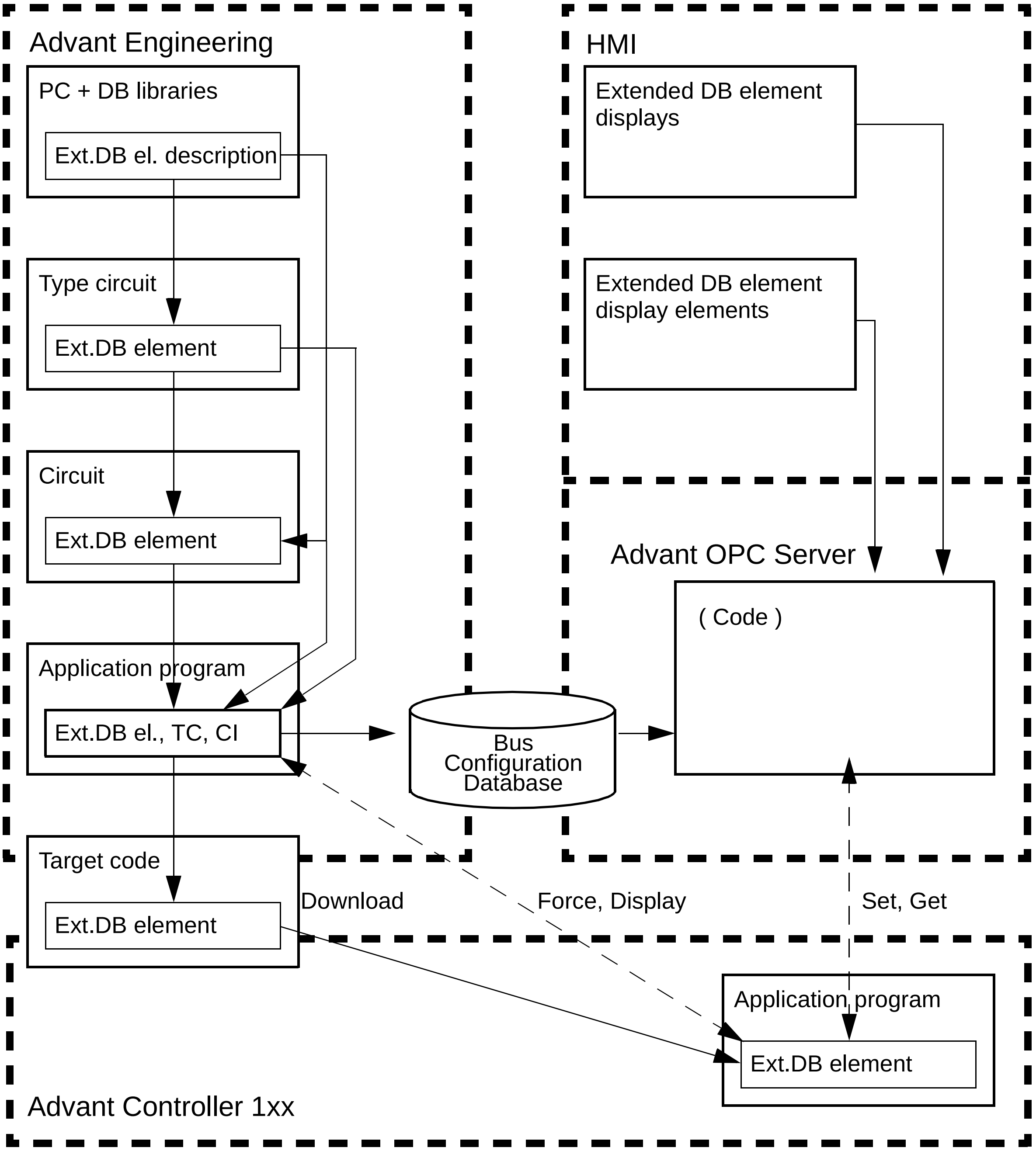


Figure 4-33. Extended DB Element, System Components with Advant OPC Server

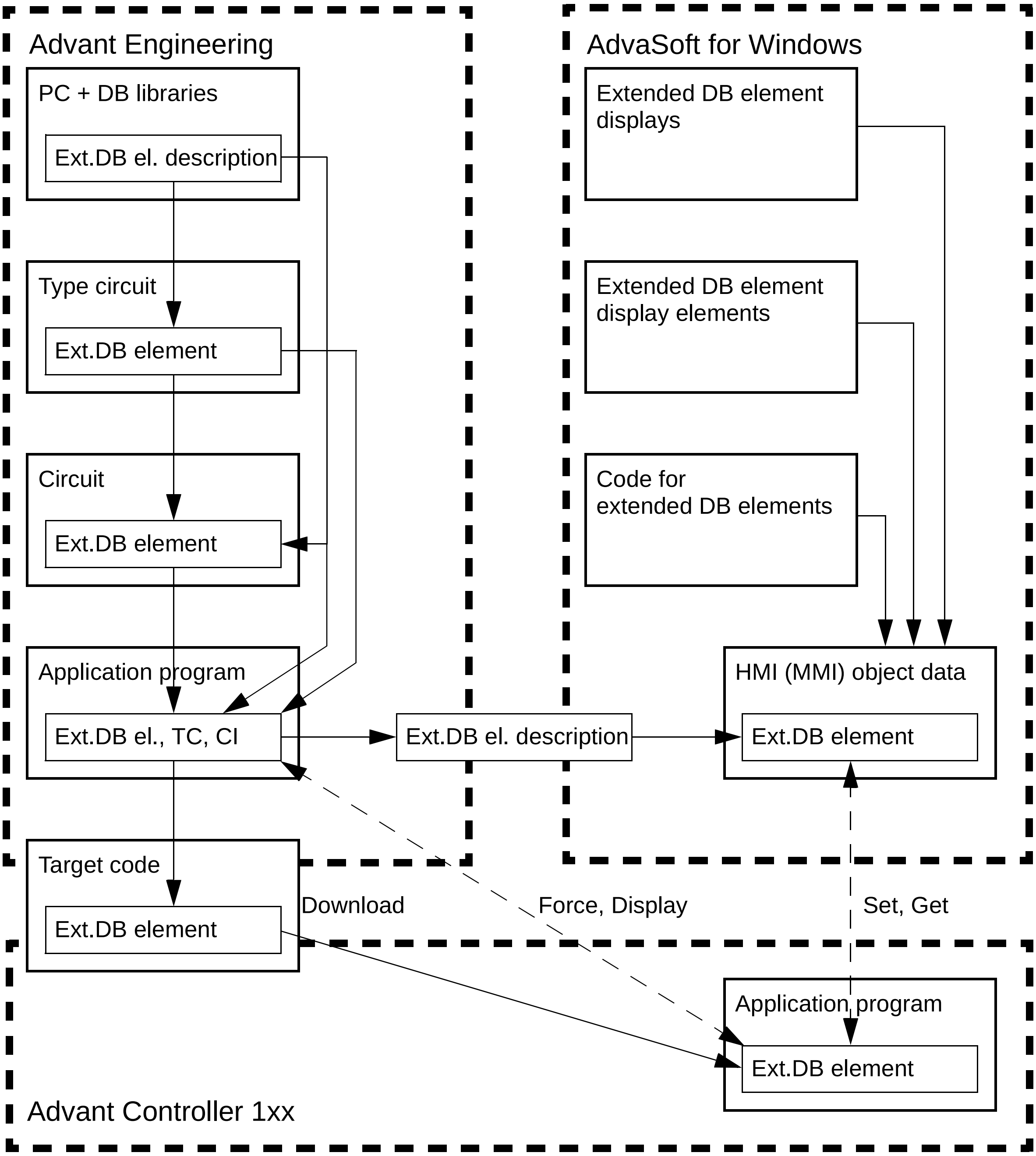


Figure 4-34. Extended DB Element, System Components with AdvaSoft for Windows

4.13.2 Extended DB Element Communication Between Stations

Extended DB elements are very versatile:

- Extended DB elements allow transfer of I/O signals or signals generated by PC programs (for example, output by PC elements) to HMI system and AC 110, 160, or derivative.
- They also allow transfer of commands and calculated data from HMI system to AC 110, 160, or derivative.
- In case of AC 160, extended DB elements support also data exchange between redundant CPUs within AC 160 stations via MDAT elements.

The following figure shows how the AIS extended DB element can be used to access and propagate an analog signal in different AC 110, 160, or derivative stations:

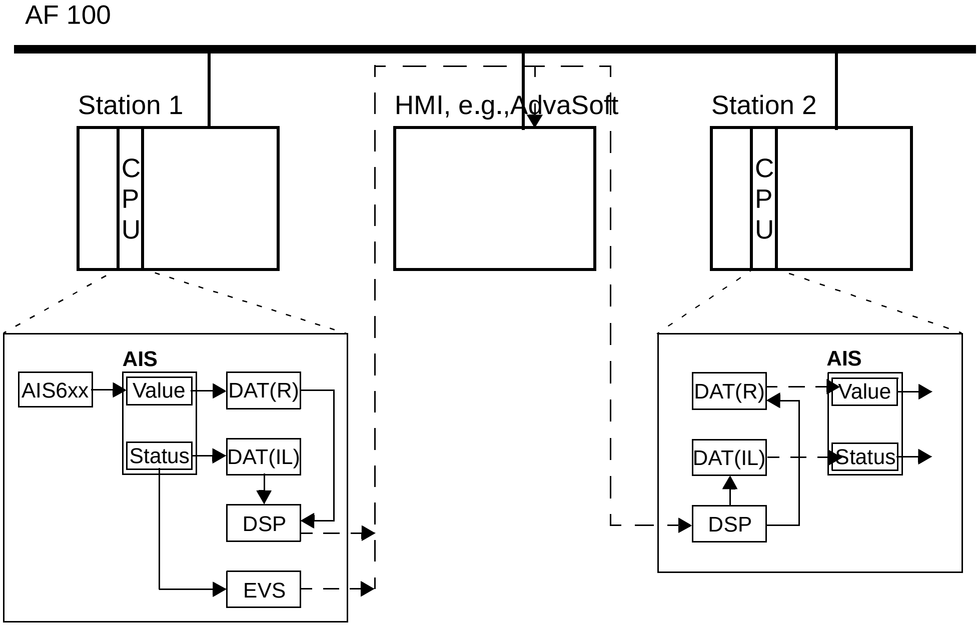


Figure 4-35. Extended DB Element Referencing HW Element

The AIS in Station 1 is configured to fetch the value from a hardware input channel and send it via DS/DAT communication. Here, the extended DB element is referenced also by an EVS(S) element that provides the event queue. DSP, DAT(R), DAT(IL), and EVS are created automatically at the creation of AIS. AIS6xx, for example, AIS630, has to be created explicitly.

The AIS in Station 2 is configured to receive the value from Station 1 via DS/DAT communication. DSP, DAT(R), and DAT(IL) are created automatically at the creation of AIS.

4.13.3 Selection of Signal Sources

Each extended DB element has a SOURCE terminal. By this terminal you define whether the extended DB element receives its “value” from the hardware, from a PC element, from another node, or from a HMI (MMI).

4.13.3.1 Overview of Source Selection

The following table shows the functions and features of extended DB elements, how the settings in the SOURCE terminal affect their functions, and, in fact, for which purposes the different extended DB element types can be used.

Table 4-14. Function and Features of Extended DB Elements

Type	**Source**	**Function**	**Features**										
			Channel Input	PC Input	Channel Output	Force Logic	Limits	Events	Ack. Logic	Send DS/DAT	Receive DS/DAT	Value blocking by MMI	Alarm blocking by MMI
AIS	HW	Read from input channel and send via DS/DAT	x			x	(x)	(x)	x	(x)		x	x
	PC	Read from PC element and send via DS/DAT.		x		x	(x)	(x)	x	(x)		x	x
	MMI	Read from HMI (e.g.,AdvaSoft) and send via DS/DAT.				x	(x)	(x)		(x)			
	RECEIVE	Receive a value from DS/DAT communication.				x					x		
DIS	HW	Read from input channel and send via DS/DAT	x			x		(x)	x	(x)		x	x
	PC	Read from PC element and send via DS/DAT.		x		x		(x)	x	(x)		x	x
	MMI	Read from HMI (e.g.,AdvaSoft) and send via DS/DAT.				x				(x)			
	RECEIVE	Receive a value from DS/DAT.				x					x		
AOS	PC	Read from PC element, write to output channel, and send via DS/DAT.			x	x	(x)	(x)	x	(x)		x	
	MMI	Read from HMI(e.g.,AdvaSoft), write to output channel, and send via DS/DAT.			x	x	(x)	(x)		(x)			
	RECEIVE	Receive a value from DS/DAT communication and write it to output channel.			x	x					x		
DOS	PC	Read from PC element, write to output channel, and send via DS/DAT.			x	x		(x)	x	(x)		x	x
	MMI	Read from HMI(e.g.,AdvaSoft), write to output channel, and send via DS/DAT.			x	x				(x)			
	RECEIVE	Receive a value from DS/DAT communication and write it to output channel.			x	x					x		

Table 4-14. Function and Features of Extended DB Elements (Continued)

Type	Source	Function	Features										
			Channel Input	PC Input	Channel Output	Force Logic	Limits	Events	Ack. Logic	Send DS/DAT	Receive DS/DAT	Value blocking by MMI	Alarm blocking by MMI
MB, MBS	PC	Read from PC element, store, and send via DS/DAT communication.				x		(x)		(x)		x	
	MMI	Read from HMI(e.g.,AdvaSoft), store, and send via DS/DAT communication.				x				(x)			
	RECEIVE	Receive a value from DS/DAT and store it.				x					x		
MI, MIL, MR	PC	Read from PC element, store, and send via DS/DAT communication.				x				(x)		x	
	MMI	Read from HMI(e.g.,AdvaSoft), store, and send via DS/DAT communication.				x				(x)			
	RECEIVE	Receive a value from DS/DAT and store it.				x					x		

Legend: x stands for a feature available for the given extended DB element with a given source.
(x) stands for an available feature that can be disabled at configuration time in the FCB.

AC 160 supports MDAT communication in all cases where DS/DAT communication is supported.

4.13.3.2 Selecting HW (Channel Element) as Source

To select a channel element as source for an extended DB element, set the SOURCE terminal to HW and set the HW_REF terminal to the channel element.
You can select HW as source only in case of input extended DB elements AIS and DIS.

FCB sets the corresponding SW_REF terminal of the hardware element automatically to the connected extended DB element:

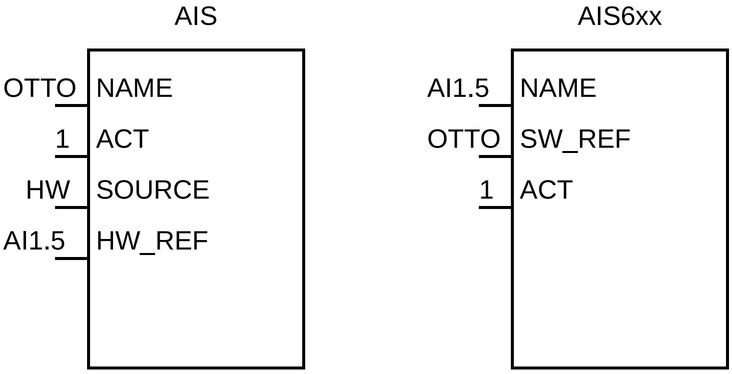


Figure 4-36. HW as Source of Extended DB Element

The module and channel DB elements of the corresponding I/O modules are configured separately as usual.

NOTE

> The HW_REF, a reference terminal like REF*i* of DSP, can also be used for navigation during DB editing with FCB. If you click on the HW_REF terminal with the right mouse button, the **Edit DB Element** dialog box in the **DB Section** window will display the referenced element.

If specified so, the AIS element converts the “raw” value received from the input channel element from percent or physical units to engineering units.
The AIS element writes the direct or converted value to the CALC_VALUE terminal, tests it for limits if specified so, and propagates it to the VALUE terminal.
The DIS element writes the value received from the input channel to the CALC_VALUE terminal and propagates it to the VALUE terminal.
Together with updating VALUE, AIS and DIS write the current status to the STATUS terminal.

4.13.3.3 Selecting PC (PC Element) as Source

To select a PC element as source for an extended DB element, set the SOURCE terminal to PC.

Connect the output of the PC element to the CALC_VAL terminal of the extended DB element.

PC as Source for Input Extended DB Elements AIS and DIS

The AIS element tests the value on the CALC_VALUE terminal for limits if specified so, and writes the value supplied by the connected PC element output (instead of, for example, the value of some input channel element) to the VALUE terminal.

The DIS element propagates the value on the CALC_VALUE terminal (instead of, for example, the value of some input channel element) to the VALUE terminal.

PC as Source for Output Extended DB Elements AOS and DOS

The AOS element tests the CALC_VALUE terminal for limits if specified so, writes the limited or direct value to the VALUE terminal, and often writes it also to some output channel element AOS6xx, after conversion from engineering units to percent or physical units if specified so.

See an example in the following figure:

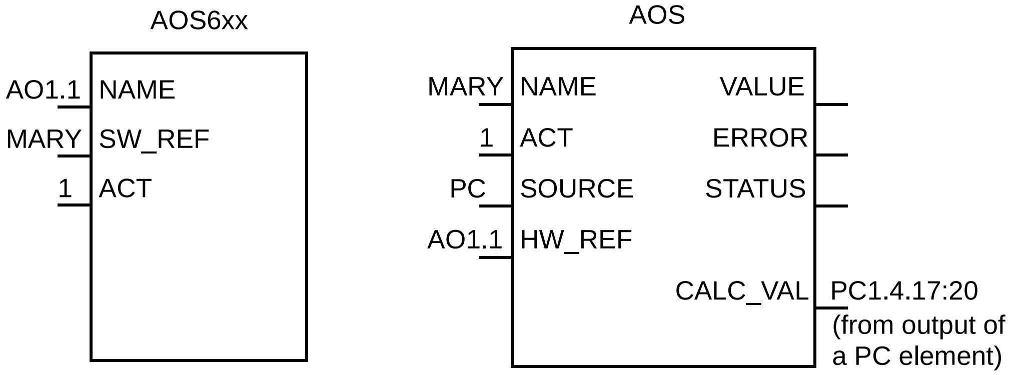


Figure 4-37. PC as Source of Extended DB Element

The DOS element propagates the value on the CALC_VALUE terminal to the VALUE terminal, and often writes it also to some output channel element DOS6xx.

If an output channel element is connected as sink by an appropriate reference on the HW_REF terminal, the extended DB element writes the current status to the STATUS terminal together with updating VALUE.

PC as Source for Memory Extended DB Elements MB, MI, MIL, and MR

The extended DB element exhibits the value supplied by the PC element output on the CALC_VALUE terminal, stores and writes it to the VALUE terminal.

4.13.3.4 Selecting DSP/DAT as Source

To connect an extended DB element to DAT elements for receiving data from AF 100, set the SOURCE terminal to RECEIVE, set also the DS_COM terminal to RECEIVE, and fill the DSP_STAT, DSP_IDNT, DSP_CYCL, and DSP_REF terminals appropriately.

The value of DSP_STAT defines the STATION of the DSP to be used.
The value of DSP_IDNT defines the IDNT of the DSP to be used.
The value of DSP_CYCL defines transmission cycle time of the DSP to be used.
The value of DSP_REF defines the order number of reference to DAT (or DATs) in the DSP to be used.

Then FCB creates the necessary DSP and DAT elements automatically and references the DAT elements in DSP.

In case of the analog extended DB elements AIS and AOS, FCB creates and references a DAT(R) for the value and a DAT(IL) for status.

In case of the digital extended DB elements DIS and DOS, FCB creates and references a DAT(I) for the status. The value is propagated as one bit of the status.

In case of the memory extended DB elements MB, MI, MIL, and MR, FCB creates and references the appropriate DAT for the value.

See also Section 4.13.6.2, Creation of DSP/DAT/EVS(S) Elements.

DSP/DAT as Source for Input Extended DB Elements AIS and DIS

The extended DB element exhibits the value supplied by AF 100 via DAT on the CALC_VALUE terminal and writes it to the VALUE terminal.

Together with updating VALUE, the extended DB element receives also the pertaining status by AF 100 via DAT and writes it to the STATUS terminal.

DSP/DAT as Source for Output Extended DB Elements AOS and DOS

The extended DB element exhibits the value supplied by AF 100 via DAT on the CALC_VALUE terminal and writes it to the VALUE terminal.

DOS often writes it also to some output channel element DOS6xx.

AOS often writes it also to some output channel element AOS6xx,after conversion to percent or physical units if specified so.

If an output channel element is connected as sink by an appropriate reference on the HW_REF terminal, the extended DB element writes the current status to the STATUS terminal together with updating VALUE.

See an example in the following figure:

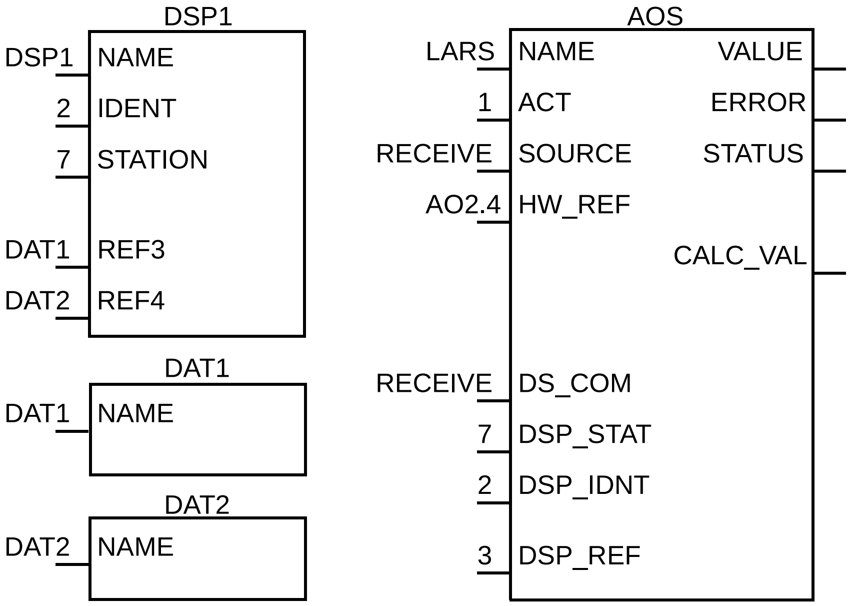


Figure 4-38. DSP/DAT as Source of Extended DB Element

DSP/DAT as Source for Memory Extended DB Elements MB, MI, MIL, and MR

The extended DB element exhibits the value supplied by AF 100 via DAT on the CALC_VALUE terminal, stores and writes it to the VALUE terminal.

4.13.3.5 Selecting MDAT as Source for Multiple CPU Support

Available only for systems that support multiple CPUs, such as AC 160. Communication interfaces like CI626 offer common data area to support data communication between redundant CPUs. The access to this data area realized by MDAT DB elements.

To connect an extended DB element to MDAT elements to receive data originating from a redundant CPU in the same station, set the SOURCE terminal to RECEIVE, set also the MDAT_COM terminal to RECEIVE, and fill the MDAT_ID terminal appropriately.

The value of MDAT_ID defines the order number of MDAT (or MDATs) to be used.

NOTE

Do not set the DS_COM terminal to RECEIVE!

Then FCB creates the necessary MDAT elements automatically.

In case of analog extended DB elements AIS and AOS, FCB creates and references an MDAT(R) for the value and an MDAT(IL) for status.

In case of digital extended DB elements DIS and DOS, FCB creates and references an MDAT(I) for the status. The value is propagated as one bit of the status.

In case of memory extended DB elements MB, MI, MIL, and MR, FCB creates and references the appropriate MDAT for the value.

See also Section 4.13.6.2, Creation of DSP/DAT/EVS(S) Elements.

MDAT as Source for Input Extended DB Elements AIS and DIS

The extended DB element exhibits the value supplied by the MDAT element on the CALC_VALUE terminal and propagates it to the VALUE terminal (instead of, for example, the value of some input channel element).

MDAT as Source for Output Extended DB Elements AOS and DOS

The extended DB element exhibits the value supplied by the MDAT element on the CALC_VALUE terminal and propagates it to the VALUE terminal and often to some output channel element (AOS6xx). See an example in the following figure:

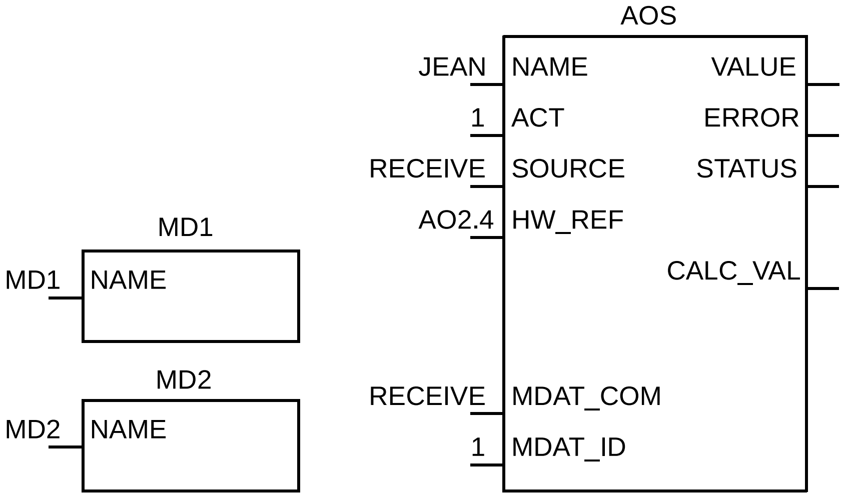


Figure 4-39. MDAT as Source of Extended DB Element

MDAT as Source for Memory Extended DB Elements MB, MI, MIL, and MR

The extended DB element exhibits the value supplied by the MDAT element via DAT on the CALC_VALUE terminal, stores and writes it to the VALUE terminal.

4.13.3.6 Selecting HMI (MMI) as Source

To connect an extended DB element to receive data from a Human-Machine Interface system (HMI, substitute for MMI [Man-Machine Interface], such as AdvaSoft for Windows or AF100 OPC), via AF 100, set the SOURCE terminal to MMI.

NOTE

The HMI system transfers the value via AF 100.
This kind of transfer does not require any DB element.

The HMI data overrides the value on the VALUE output. This new value propagates to the connected sinks.
HMI has to transfer values in engineering units. No conversion is possible.
See example in the following figure:

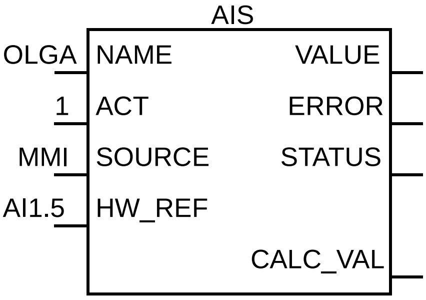


Figure 4-40. HMI (MMI) as Source of Extended DB Element

4.13.3.7 Engineering Station as Source

The Advant Station 100 Series Engineering Station can also be source of value for extended DB elements. For that use the force function of Function Chart Builder.

Start FCB for the node, connect the node, set on-line mode, select the appropriate signals for forcing, set appropriate force values, and activate force values.
It is not necessary to change setting of the SOURCE terminal.
Forcing from FCB overrides any other signal source.

NOTE

FCB transfers the value via AF 100.
This kind of transfer does not require any DB element, either.

The extended DB element overrides the value on the VALUE output by the force value supplied by FCB and propagates this new value to the connected sinks.
FCB transfers force values in engineering units. No conversion is necessary.

4.13.4 Processing in Extended DB Elements

The controller processes the extended DB elements by cyclical tasks consisting of the following phases:

1. Input - All the data is fetched from external sources.
2. Processing - The data itself is processed.
3. Output - The generated data is transferred to external consumers.

The task runs with a cycle time equal to the shortest SCANT specified on the extended DB elements. The execution is optimized, the extended DB elements are processed only when it is necessary due to their own SCANT. The extended DB elements are grouped by scan times.

If the user specifies a scan time greater than 4096 ms for an extended DB element and either exports or imports it via AF 100, the dataset scan time is set to 4096 ms. This is the largest value allowed for data sets.

The processing generally covers the following:

- Conversion of raw values (values in percent or physical units) delivered by some AIS6xx analog input channel elements to values in engineering units on the CALC_VAL terminal
- Checking of limit violation by the value
- Limitation of value to allowed violation (only AOS)
- Transfer from CALC_VAL to VALUE terminals
- Conversion of values in engineering units on the CALC_VAL terminal to values in percent or physical units as required by some AOS6xx analog output channel elements
- Event generation.

4.13.4.1 Transfer Between CALC_VAL and VALUE

Each extended DB element has the following two terminals:

- CALC_VALUE
  It is considered preferably as input terminal, though it can be read and written.
  It contains some input value generally in engineering units.
  If the source is an analog input channel element and you specify no conversion on the controller by setting SCALING to NONE or MMI, CALC_VAL contains the “raw” value which is expressed in percent or in physical units. For example, AI610: 0 - 100%; AI620: 0 - 100% or -100 - 100%; AI625: 0 - 100%; AI630: °C or Ω; AI635: °C or mV.
- VALUE
  It is considered preferably as output terminal, though it can be read and written.
  It contains some output value generally in engineering units.
  The controller copies here the value of CALC_VALUE simply, or in case of AOS and if you specified limit checking, after limitation.
  CALC_VAL may (and has to) contain values expressed in percent or in physical units if the sink is an analog output channel element and you specify no conversion on the controller by setting SCALING to NONE or MMI. For example, AO610: 0 - 100%; AI650: 0 - 100% or -100 - 100%.

Figure 4-41 shows the data flow between CALC_VAL and VALUE terminals of extended DB elements as seen by the user. Sources, sinks, and actions are pointed out.
The data flow is valid for the case of ACT=1 and HW_PCDIA=NO.

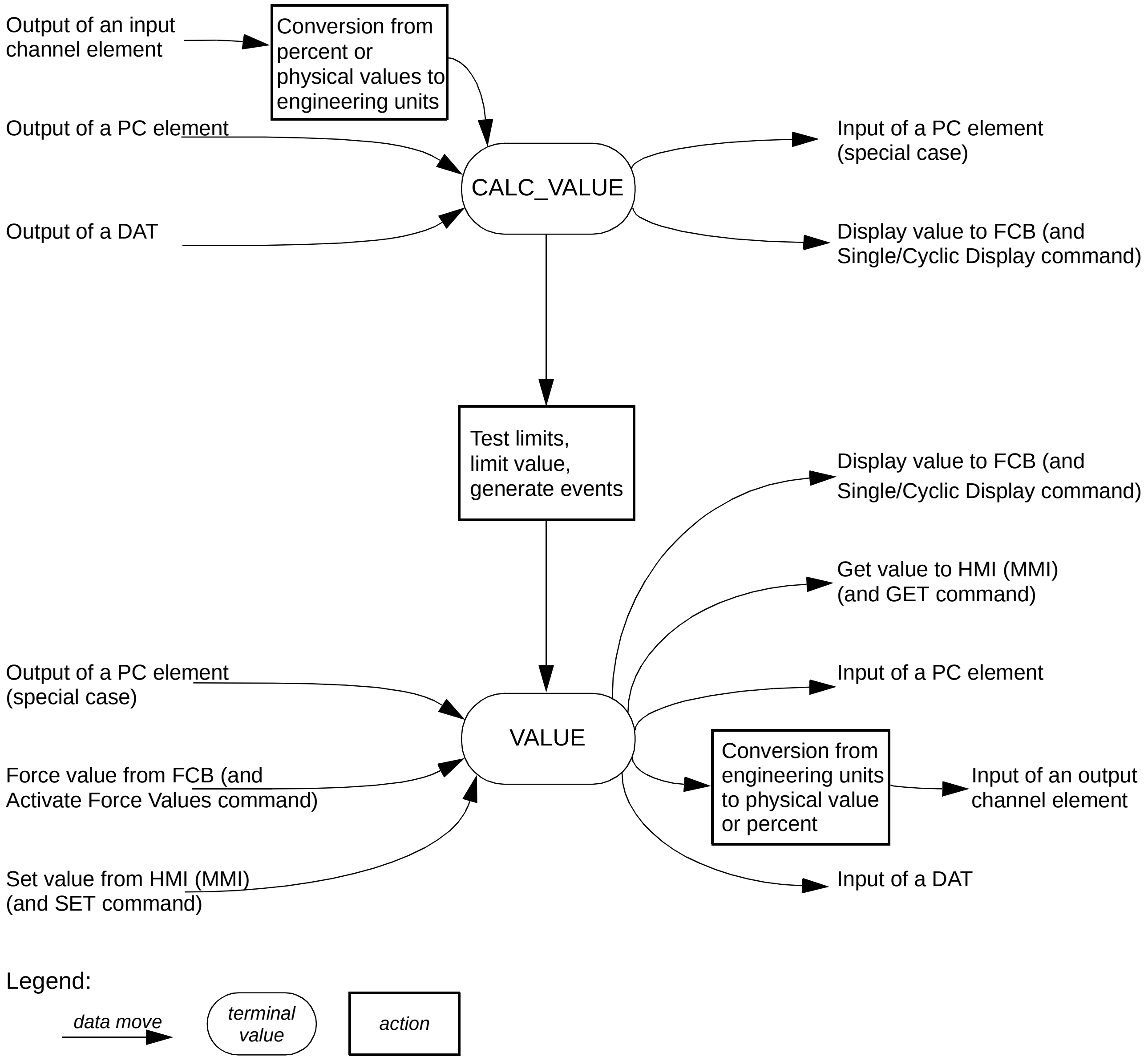

Output of an input channel element
Conversion from percent or physical values to engineering units
Output of a PC element
CALC_VALUE
Output of a DAT
Input of a PC element (special case)
Display value to FCB (and Single/Cyclic Display command)
Test limits, limit value, generate events
Display value to FCB (and Single/Cyclic Display command)
Get value to HMI (MMI) (and GET command)
Input of a PC element
Output of a PC element (special case)
VALUE
Force value from FCB (and Activate Force Values command)
Set value from HMI (MMI) (and SET command)
Conversion from engineering units to physical value or percent
Input of an output channel element
Input of a DAT
Legend:
data move
terminal value
action

Figure 4-41. Extended DB Element Data Flow

Connections to and from PC elements, denoted by “special case”, are allowed but not recommended non-standard way of extended DB element usage.

4.13.4.2 Status Handling with Extended DB Elements

In case of the analog extended DB elements AIS and AOS, FCB creates and references not only a DAT(R) for the value but also a DAT(IL) for status.
In case of the digital extended DB elements DIS and DOS, FCB creates and references a DAT(I) for the status. The value is represented by one bit of the status.

In case of analog extended DB elements AIS and AOS and if multiple CPU support is specified, FCB creates and references an MDAT(R) for the value and also an MDAT(IL) for status.
In case of digital extended DB elements DIS and DOS and if multiple CPU support is specified, FCB creates and references an MDAT(I) for the status. The value is represented by one bit of the status.

Together with transferring VALUE, extended DB elements connected via DATs and AF 100 transfer also the status. Together with updating VALUE, extended DB elements write the current status to the STATUS terminal.

4.13.4.3 Event Handling with Extended DB Elements

To allow an extended DB element to generate events, set its EV_COM terminal to SEND.
Fill also the EVS_IDNT and EVS_REF terminals appropriately.

Then FCB creates the necessary EVS(S) element automatically.

The extended DB elements can generate events in the following cases:

- Operational changes
- Error conditions.

Examples

1. The following events are generated by an **AIS** element:
    - Limit violation (HI2, HI1, LO1, LO2)
    - AIS6xx error
    - Alarm blocking
    - Forcing
    - Acknowledge.
2. The following events are generated by a **DOS** element:
    - DOS6xx error
    - Forcing
    - Acknowledge
    - Value change.
3. The following event is generated by an **MB** element:
    - Value change (any of the 32 bit)
    - Acknowledge.

4.13.4.4 Operational Mode Change Events

An operational mode change event will be generated in the following cases:

Table 4-15. Operational Mode Change Events

EVS_STATUS	UPDATE_BLK_CMD	ALARM_BLK_CMD	Limit Change	Acknowledge	Event
0	x	x	x	x	No event
1	Toggle	x	x	x	Mode change event. UPDATE_BLK status bit is set by AdvaSoft command.
1	x	Toggle		x	Mode change event. ALARM_BLK status bit is set by AdvaSoft command.
1	x	x	Change(s)	x	Mode change event. Specific change bit is set to 1.
1	x	x	x	0 -> 1	Mode change event. Status bits ALARM_UNACK and ACK_CMD are reset.

A limit violation event will be always generated if the following 3 conditions are met:

- Limit checking is enabled.
- A limit is violated or a violation ends.
- The ALARM_BLK bit of status is in 0 state in the controller.

4.13.4.5 Error Condition Events

An error condition event will be generated in the following cases:

Table 4-16. Error Condition Events

EVS_ERR	Source	HWError	CALC_Err	ERROR	Event
0	x	x	x	x	No event
1	HW	1 -> 0	x	FALSE	Yes
1	HW	0 -> 1		TRUE	Yes
1	PCx	x	1 -> 0	FALSE	Yes
1	PCx	x	0 -> 1	TRUE	Yes
1	MMI			FALSE	No event
1	RECEIVE			Sender	No event

4.13.5 Connection to Signal Sinks

4.13.5.1 Connecting Extended DB Elements to HW (Channel Element) Sinks

To connect an output extended DB element to a HW sink, set the HW_REF terminal to the appropriate output channel element.

You can select a HW sink only in case of output extended DB elements AOS and DOS.

The output extended DB elements AOS tests the CALC_VALUE terminal for limits if specified so, writes the limited or direct value to the VALUE terminal, converts it to percent or physical units from engineering units if specified so, and propagates it to the output channel element AOS6xx.

FCB sets the corresponding SW_REF terminal of the hardware element automatically to the connected extended DB element:

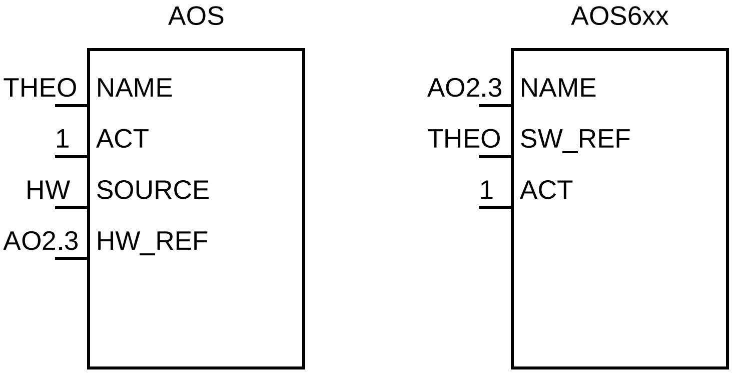


Figure 4-42. HW as Sink of Extended DB Element

The module and channel DB elements of the corresponding I/O modules are configured separately as usual.

NOTE

The HW_REF, a reference terminal like REF*i* of DSP, can also be used for navigation during DB editing with FCB. If you click on the HW_REF terminal with the right mouse button, the **Edit DB Element** dialog box in the **DB Section** window will display the referenced element.

The output extended DB elements DOS copies the value of the CALC_VALUE terminal to the VALUE terminal, and propagates it to the output channel element DOS6xx.

Together with updating VALUE, AOS and DOS write the current status to the STATUS terminal.

4.13.5.2 PC Element Inputs as Extended DB Element Sinks

To connect an output of an extended DB element to an input of a PC element, use the FCB in the same way as in the case of other DB elements.

See also Section 4.13.6.1, Connecting Extended DB Elements to PC Elements.

4.13.5.3 Selecting DSP/DAT as Sink

To connect DAT elements to an extended DB element for sending data via AF 100, set the DS_COM terminal to SEND and fill the DSP_STAT, DSP_IDNT, DSP_CYCL, and DSP_REF terminals appropriately.

The value of DSP_STAT defines the sending station. In this case, it is the number of the own station.
The value of DSP_IDNT defines the IDNT of the involved DSP.
The value of DSP_CYCL defines transmission cycle time of the involved DSP.
The value of DSP_REF defines the order number of reference to DAT (or DATs) in the involved DSP.

Then FCB creates the necessary DSP and DAT elements automatically and references the DAT elements in DSP.

In case of the analog extended DB elements AIS and AOS, FCB creates and references a DAT(R) for the value and a DAT(IL) for status.

In case of the digital extended DB elements DIS and DOS, FCB creates and references a DAT(I) for the status. The value is propagated as one bit of the status.

In case of the memory extended DB elements MB, MI, MIL, and MR, FCB creates and references the appropriate DAT for the value.

See also Section 4.13.6.2, Creation of DSP/DAT/EVS(S) Elements.

The values on the VALUE and STATUS terminals of the extended DB element are sent by the involved DSP and DATs to AF 100.

See an example in Figure 4-43. The AIS in the example references also an AIS6xx channel element but the latter is not shown in the figure.

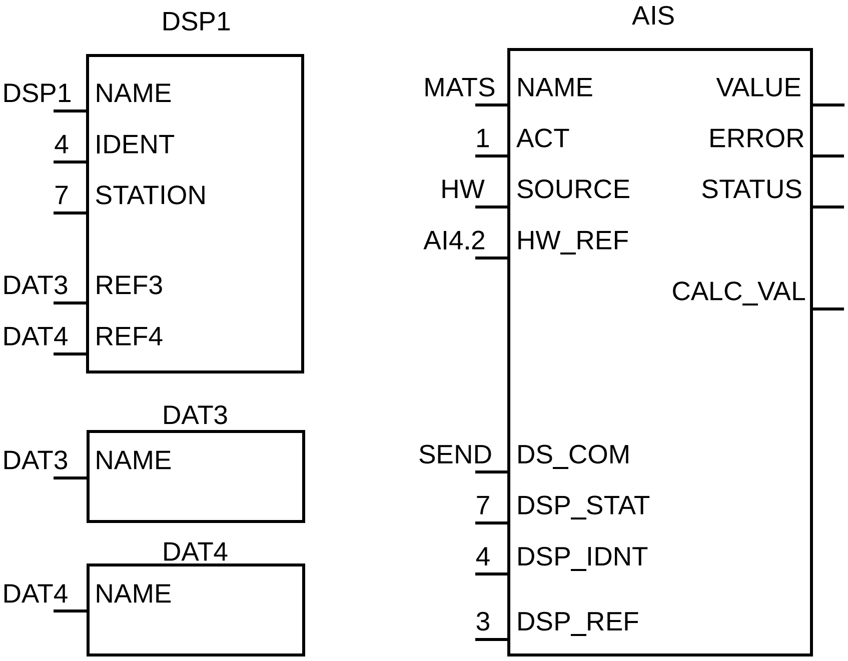


Figure 4-43. DSP/DAT as Sink of Extended DB Element

4.13.5.4 Selecting MDAT as Sink for Multiple CPU Support

Available only for systems that support multiple CPUs, such as AC 160. Communication interfaces like CI626 offer common data area to support data communication between redundant CPUs. The access to this data area realized by MDAT DB elements.

To connect MDAT elements to an extended DB element for sending data from a redundant CPU, set the MDAT_COM terminal to SEND and fill the MDAT_ID terminal appropriately.

The value of MDAT_ID defines the order number of involved MDAT (or MDATs).

Then FCB creates the necessary MDAT elements automatically.

In case of analog extended DB elements AIS and AOS, FCB creates and references an MDAT(R) for the value and an MDAT(IL) for status.

In case of digital extended DB elements DIS and DOS, FCB creates and references an MDAT(I) for the status. The value is propagated as one bit of the status.

In case of memory extended DB elements MB, MI, MIL, and MR, FCB creates and references the appropriate MDAT for the value.
See also Section 4.13.6.2, Creation of DSP/DAT/EVS(S) Elements.

The values on the VALUE and STATUS terminals of the extended DB element are sent by the involved MDATs.
See an example in Figure 4-44. The AIS in the example references also an AIS6xx channel element but the latter is not shown in the figure.

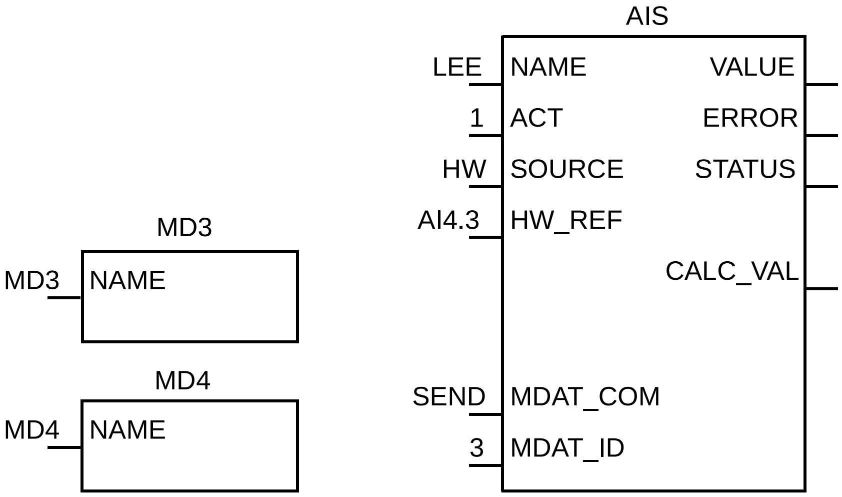


Figure 4-44. MDAT as Sink of Extended DB Element

4.13.5.5 HMI (MMI) as Sink

HMI (MMI) transfers the value on the VALUE terminal of the extended DB element via AF 100.

This kind of transfer does not require any DB element.

HMI gets the value in engineering units. No conversion is necessary.

4.13.5.6 Engineering Station as Sink

The Advant Station 100 Series Engineering Station can also be sink of value for extended DB elements. For that use the display function of Function Chart Builder.

Start FCB for the node, connect the node, set on-line mode, define the appropriate signals for monitoring, and activate the monitoring by any of the display commands.

FCB transfers the value via AF 100.
This kind of transfer does not require any DB element, either.

4.13.6 Miscellaneous

4.13.6.1 Connecting Extended DB Elements to PC Elements

If you want to connect a PC element terminal to an I/O signal referenced by an extended DB element, **connect the appropriate terminal of the extended DB element** (AIS, AOS, DIS, DOS) instead of connecting directly to the channel element (AIS6xx, AOS6xx, DIS6xx, DOS6xx). This is valid for the VALUE terminal and also the limit check terminals.

The terminals of the I/O module, especially the limit check terminals, may not be consistent with the configuration of the extended DB element.

Connecting PC elements to the extended DB element instead of the channel element ensures that even an exchange of the HW DB element does not affect the application program.

4.13.6.2 Creation of DSP/DAT/EVS(S) Elements

Automatic Creation of DSP/DAT/EVS(S)

If you create an extended DB element, all necessary DSP, DAT, and EVS(S) elements are automatically created as specified by values of the DS_COM, DSP_STAT, DSP_IDNT, DSP_CYCL, DSP_REF, EV_COM, EVS_IDNT, and EVS_REF terminals. All necessary references are automatically established between the extended DB element and the DSP, DAT, and EVS(S) elements.

DATs for value and status pertaining to an analog extended DB element AIS or AOS have to be referenced by one and the same DSP element.

If the DAT(R) element for value of an analog extended DB element AIS or AOS is referenced by terminal REF*i* of a pertaining DSP where *i* is defined by the terminal DSP_REF, then the DAT(IL) element for status of the same extended DB element is referenced by terminal REF*i*+1 of the same DSP.

An analog extended DB element AIS or AOS references the MDAT(R) element MD*i* by the value *i* of terminal MDAT_ID. Then, the MDAT(IL) element for status of the same extended DB element is MD*i*+1.

Visibility of Automatically Created DSP/DAT/EVS(S)

The automatically created DB elements are invisible for you in FCB unless you allow it by the **Options | Preferences** command and the pertaining option. In the **DB Section** they are denoted by the attribute ‘X’.

The automatically created DB elements do not appear in source code.

Nevertheless, they appear in the documentation of the DB section.
Use the **Print** command of FCB.

Automatically and Manually Created Elements

DB elements DSP, DAT, and EVS(S) can also be created separately by the user.

So there are two kinds of same DB elements, automatically created ones and explicitly created ones. But only the latter are always visible in FCB and appear in the source code. Nevertheless, you should bear in mind that the former exist, too.

4.13.6.3 Acknowledge Logic of HMI (MMI) Connection

Acknowledgment is done on the basis of a single extended DB element. This means that always the entire extended DB element is acknowledged and never single signals or limits of it.

The acknowledgment state of an extended DB element is stored in and managed by the AC 110, 160, or derivative. To acknowledge an extended DB element, the HMI system, such as AdvaSoft for Windows, sends an acknowledgment command to the AC 110, 160, or derivative. The controller updates the extended DB element’s acknowledgment state and, if appropriate, generates an event message.

4.13.7 Special Terminals of Extended DB Elements

4.13.7.1 Special Terminals in Base Part of Extended DB Elements

ACT

Table 4-17. ACT Terminal

Terminal	Value	Comment
ACT	1	Extended DB element is active with all functions.
	0	Extended DB element is not active. But if HW_PC_DIA = YES, the accelerated execution of AIS, AOS; DIS, and DOS (bypass of CALC_VAL to VALUE path) remains active.

SOURCE

Table 4-18. SOURCE Terminal

Terminal	Value	Comment
SOURCE	HW	Available only for AIS and DIS. The value is fetched from the hardware, the terminal HW_REF is used to specify the AIS6xx or DIS6xx element, respectively.
	PC	The value is supplied by a PC element connected to the CALC_VAL terminal.
	RECEIVE	The value is received from DSP/DAT (via AF 100).
	MMI	The value is provided by HMI, for example, AdvaSoft for Windows.

HW_REF

Table 4-19. HW_REF Terminal

Terminal	Value	Comment
HW_REF	*reference* like AI2.7	Available for AIS, AOS, DIS, DOS. In case of AIS and DIS reference to AIS6xx or DIS6xx (hardware) channel element if the SOURCE is set to HW. In case of AOS and DOS reference to AOS6xx or DOS6xx (hardware) channel element.

VALUE

Table 4-20. VALUE Terminal

Terminal	Type	Comment
VALUE	Real for AIS, AOS, and MR	For AOS: The AOS6xx channel elements require the value in percent or physical units. If VALUE is expressed in engineering units, set SCALING=CONTROLLER to get the value converted accordingly between VALUE and the AOS6xx channel element.
	Integer for MI	
	Integer Long for MIL and MB (32 bits packed)	
	Boolean for MBS	

CALC_VAL

Table 4-21. CALC_VAL Terminal

Terminal	Type	Comment
CALC_VAL	Real for AIS, AOS, and MR	For AIS: The value is expressed in percent or physical units as the AIS6xx channel elements deliver them. If you need CALC_VAL expressed in engineering units, set SCALING=CONTROLLER, the value is converted to get the value converted accordingly between the AIS6xx channel element and CALC_VAL.
	Integer for MI	
	Integer Long for MIL and MB (32 bits packed)	
	Boolean for MBS	

STATUS

Table 4-22. STATUS Terminal

Terminal	Type	Comment
STATUS	Integer Long	Status information of the extended DB element. Only for AIS, AOS, DIS, and DOS.

CALC_ERR

Table 4-23. CALC_ERR Terminal

Terminal	Value	Comment
CALC_ERR	1	Signals an error in any of the following cases: For AOS and AIS: If SCALING=CONTROLLER and a conversion (software) error occurs. For AIS and DIS: If SOURCE is HW and a hardware error occurs. For AOS and DOS: If AOS6xx or DOS6xx is referenced by HW_REF and a hardware error occurs.
	0	

4.13.7.2 Terminals in Limit Check Part of Extended DB Elements

HI_LIM, LO_LIM (HI_LIM1, HI_LIM2, LO_LIM1, LO_LIM2),

Table 4-24. HI_LIM, LO_LIM Terminals

Terminal	Condition	Comment
HI_LIM*, LO_LIM*	LIM_SRC=HW	Available for AIS. Limit values are in percent or physical units. Limit values are checked by the AI module itself with HI_LIM… of the AIS6xx element.
	LIM_SRC=SW	Available for AIS and AOS. Limit values are in engineering units.
	LIM_SRC=NONE	No limit checks are performed.

4.13.7.3 Special Terminals in Communication Part of Extended DB Elements

DS_COM

Table 4-25. DS_COM Terminal

Terminal	Value	Comment
DS_COM	SEND	The controller sends the value, and if applicable, the status, via DSP/DAT to AF 100.
	RECEIVE	The controller receives the value on the CALC_VAL terminal, and if applicable, also the status via DSP/DAT from AF 100. SOURCE has to be set to RECEIVE.
	NONE	No DSP/DAT communication is done.

DSP_STAT

Table 4-26. DSP_STAT Terminal

Terminal	Value	Comment
DSP_STAT	1 - 80	STATION of DSP Remember it does not necessarily reflect the own physical station address. Using further, “virtual” station numbers makes it possible to have more than 50 (=max IDENT) sending DSPs per station.

DSP_IDNT

Table 4-27. DSP_IDNT Terminal

Terminal	Value	Comment
DSP_IDNT	1 - 50	IDENT (identifier) of DSP

DSP_STAT and DSP_IDNT together identify a DSP element.

DSP_REF

Table 4-28. DSP_REF Terminal

Terminal	Value	Comment
DSP_REF	1 - 8	Reference order of the first pertinent DAT in the DSP. AIS, AOS occupy two DATs, one for VALUE and one for STATUS; only DSP_REF = 1 - 7 is allowed for these elements. (Recommended values are 1, 3, 5, and 7.)

EV_COM

Table 4-29. EV_COM Terminal

Terminal	Value	Comment
EV_COM	SEND	Generate events and send them via EVS.
	NONE	No event generation.

EVS_IDNT

Table 4-30. EVS_IDNT Terminal

Terminal	Value	Comment
EVS_IDNT	1 - 32	IDENT (identifier) of EVS

EVS_REF

Table 4-31. EVS_REF Terminal

Terminal	Value	Comment
EVS_REF	1 - 32	Reference order in the EVS.

EVS_IDNT and EVS_REF together identify an EVS element.

MDAT_COM and MDAT_ID are available only for controllers with CPU redundancy in a station, such as AC 160:

MDAT_COM

Table 4-32. MDAT_COM Terminal

Terminal	Value	Comment
MDAT_COM	SEND	The controller sends the value, and if applicable, the status, via MDAT.
	RECEIVE	The controller receives the value on the CALC_VAL terminal, and if applicable, also the status via MDAT. SOURCE has to be set to RECEIVE.
	NONE	No DSP/DAT communication is done.

MDAT_ID

Table 4-33. MDAT_ID Terminal

Terminal	Value	Comment
MDAT_ID	1..80	Order number *i* in the item designation MD*i* of MDAT.

4.13.7.4 Special Terminals HMI (MMI) Part of Extended DB Elements

MMI_USE

Table 4-34. MMI_USE Terminal

MMI	Value	Comment
MMI_USE	YES	The extended DB element is marked for HMI (MMI) use. The extended DB element will be public in the HMI system, such as AdvaSoft.
	NO	The extended DB element is not marked for HMI use. The extended DB element not public in HMI system.

4.13.7.5 Terminals in Special Part of Extended DB Elements

HW_PCDIA

Table 4-35. HW_PCDIA Terminal

Terminal	Value	Comment
HW_PCDIA	YES	Available only for AIS and DIS if SOURCE = HW, and AOS and DOS if HW_REF is not empty: To accelerate extended DB element execution, the extended DB elements bypass the CALC_VAL to VALUE path with the pertaining processing and the possible conversion. So the PC elements connected to the extended DB element are shorted to the (hardware) channel element. (In controller terms: the PC elements access the channel element directly.)
	NO	Normal processing and signal propagation from source to sink via CALC_VAL and VALUE.

Advantages of direct access (HW_PCDIA=YES):

- Ensures fast access for **single** consumers (CONTRMs) within one node if the extended DB element has a slower cycle time.
- No second (asynchronous) transfer cycle with double cycle time is required in the controller.

Disadvantages of direct access (HW_PCDIA=YES):

- Force function (by FCB) becomes limited and less comprehensible because forcing the extended DB element does not influence the signal between the shorted PC element and channel element. The force value is in effect only for the normal sinks of the VALUE terminal of the extended DB element.
- It depends on the used hardware channel element whether the VALUE is provided in physical or engineering units.

Advantages of normal processing and signal propagation (HW_PCDIA=NO):

- Ensures fast access for **many** consumers (CONTRMs) within one node.
- Supports the force function of FCB in a clear and uniform way.
- The signal on the VALUE terminal is always provided in engineering units.

LIM_SRC

Table 4-36. LIM_SRC Terminal

Terminal	Value	Comment
LIM_SRC	HW	Available for AIS. The limit checks are performed by the referenced channel element.
	SW	Available for AIS and AOS. The limit checks are performed by the AIS and AOS element.
	NONE	No limit checks are performed.

SCALING

Table 4-37. SCALING Terminal

Terminal	Value	Comment
SCALING	CONTROLLER	Conversion (scaling) is done by the controller. For AIS: The controller converts the “raw” values delivered by the channel element (in percents or physical units) to engineering units. For AOS: The controller converts the engineering units to values in percents or physical units as required by the channel element.
	MMI	Conversion (scaling) is done on the HMI (MMI). The controller uses values in percents or physical units as delivered or required by channel elements.
	NONE	No conversion (scaling) is done.

4.13.8 Example of Extended DB Element Configuration

The following example is a fragment of AC 110 realization of Station 1 data base (see Figure 4-35).

Its creation process is the following:

1. We created an AIS610 module and edited its AI1.1 channel element.
   AI1.1 (HUGO) will be referenced by the extended DB element.
2. We created extended DB element AIS and edited it to provide communication via DSP/DAT, HMI support, and to generate events by EVS.
3. FCB created automatically the elements DSP1, DAT1, DAT2, and EVS1. These DB elements are in principle invisible for you. Nevertheless, be aware they are created.

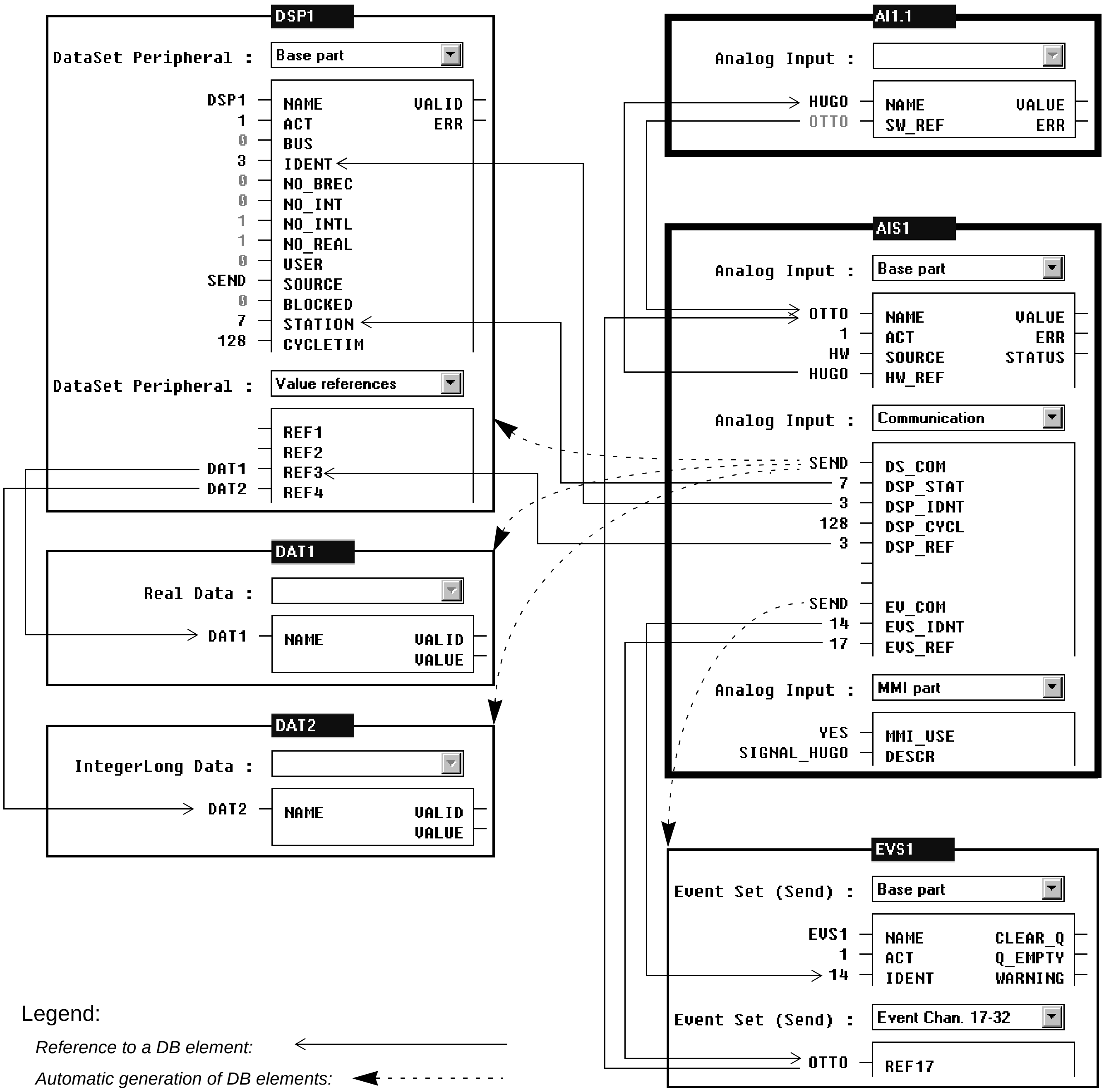


Figure 4-45. Example of Extended DB Element Configuration

4.14 PROFIBUS_DP DB Elements and Communication

See *Advant Controller 110 PROFIBUS-DP User's Guide* for description of PROFIBUS-DP communication, PROFIBUS-DP DB elements representing compact slaves, modular slaves, modules, and signals.

See *PROFIBUS Library Editor User's Guide* for their configuration with the PROFIBUS Element Library tool.

Chapter 5 Programming

5.1 General

5.1.1 PC Program

The program part of a user application program describes the control algorithm and the control strategy. It contains the required PC elements, their interconnections, and the connections to the DB elements.

This part of the user application program is called the PC program.
It consists of the PCPGM header element and subordinated structure and basic PC elements.

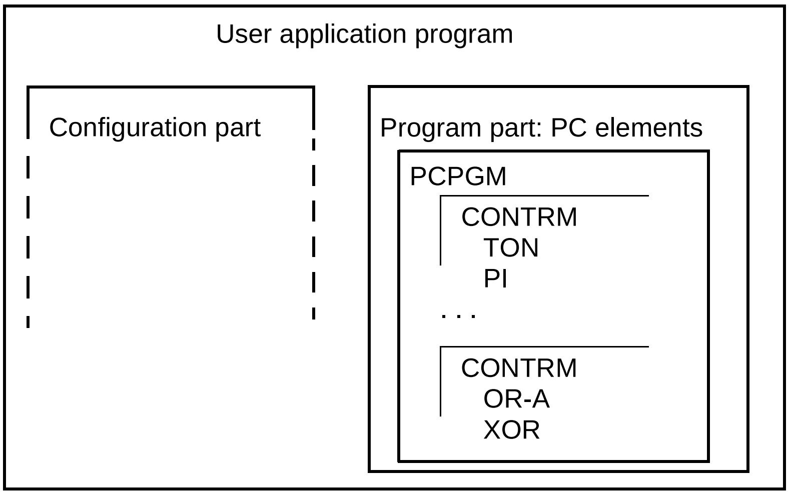


Figure 5-1. Program Part Within User Application Program

5.1.2 PC Elements in General

The fundamental unit of the PC part is the PC element.

The PC elements can be divided into two groups:

- Basic elements (in some documents also denoted as function elements)
  The implementation of a complete function
  (for example, a counter element, a delay element, …)
- Structure elements
  A header for a PC module.

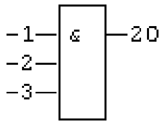


Figure 5-2. Example of a PC Element in Graphical Representation

PC elements may have input and output terminals to interconnect them and to connect them to DB elements or constants. Each element is represented in a PC diagram by a graphic symbol designed to describe the function of the element.

For a description of PC elements, see Section 5.4, PC Elements.

5.1.3 PC Diagram

The PC diagram is the usual representation of a PC program. The PC diagram is a graphical representation of the PC elements and their interconnections.

For creation and modification of PC elements and their interconnections, the engineering stations contain an editor in PC diagram representation (function chart representation).

For documentation, PC diagrams can be generated by the engineering station.

5.1.4 Executable Unit

A PCPGM on an Advant Controller 100 Series system is divided into executable units. Every executable unit acts with its own scheduling parameters, for example, with its own cycle time.

PC elements CONTRM, MASTER, SEQ, and PCPGM—together with their subordinate PC elements—form executable units.

For a description of PC elements describing executable units, see Section 5.4, PC Elements.

5.2 Data Types

PC elements handle different data types, such as logical (Boolean) variables, integers, real numbers, and so on.

A number of data types with specific properties and a range of values have been defined. The data exchange between the PC elements and the DB elements is based on standardized data types. The standardization of data types permits an automatic validity check of the format and values of data during program entry.

The following data types are used in ABB Advant Controller systems:

B	Boolean (logical 1 or 0)
I	Integer, 16 bit
IL	Integer, 32 bit
R	Real numbers
T	Time

TR	Time, expressed as a real number
A	Array, for example, a character string
G	Group data (several data of the same data type).

If the result of a calculation within an element exceeds the legal range for the data type concerned, the value at the affected output is set equal to this limit value.

Data types have been established for the call parameters and terminals of each element. For certain elements, however, the data type for some terminals may be selected with a call parameter (see Section 5.4.1.5, Call Parameters).

Many elements, for example, time delays and counters, have different data types for control signal terminals and for data terminals:

The delay element in Figure 5-3 has data type Boolean (B) at input I and output O. The input TD (Time Delay) and the output TE (Time Elapsed), on the other hand, are of data type Time (T).

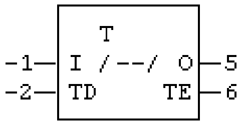


Figure 5-3. Delay Element TON

The counter in Figure 5-4 uses Boolean control signals and Integers (I or IL) for data input and output.

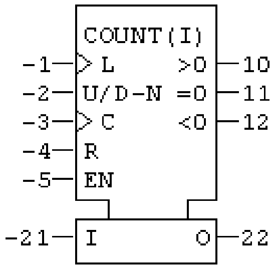


Figure 5-4. Counter Element COUNT

Each data type has a specified entry and presentation format. When you enter data via an engineering station, the format and value of the entered data is checked before it is converted to internal format.

5.2.1 Boolean

A Boolean (logical) variable can be either 1 or 0. The Boolean data type is used primarily for all logical elements, but it is also used for control signals, for example, for data handling elements.

The AND gate in Figure 5-5 has Boolean type data for all inputs and outputs.

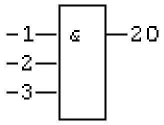


Figure 5-5. PC Element AND with Three Inputs

The multiplexer in Figure 5-6 uses Boolean values for control and for fault indication.

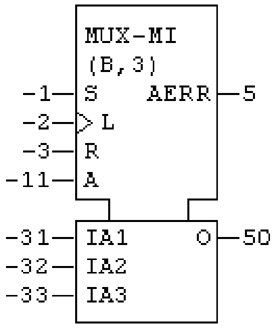


Figure 5-6. Multiplexer MUX-MI (B,3)

Entry and Presentation Format on Engineering Stations

The entry and presentation format on engineering stations is:

1	corresponding to true
0	corresponding to false.

Advant Controller 100 Series Internal Format

Internally, Boolean is stored as two bytes with:

FF00	(hexadecimal) corresponding to true
00FF	(hexadecimal) corresponding to false.

5.2.2 Integer

Two data types are available to represent integers:

I	with the range	-32768 to 32767	$(-2^{15} .. 2^{15}-1)$
IL	with the range	-2147483648 to 2147483647	$(-2^{31} .. 2^{31}-1)$

An integer (I or IL) can be positive, zero, or negative. If no sign is specified, the number is considered positive.

The counter in Figure 5-7 is an example of the use of integers. The counter can be used with both types of integers. Which one is used in a particular case is determined by a call parameter at the time the element is created.

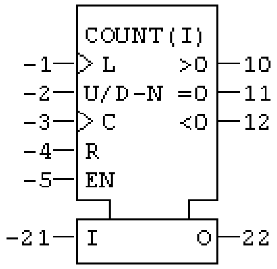


Figure 5-7. PC Element COUNT

Another example of elements that can handle both integer types is the comparator COMP-I in Figure 5-8. The outputs of this element have Boolean values and the inputs have integer values, I or IL depending on the call parameter C1.

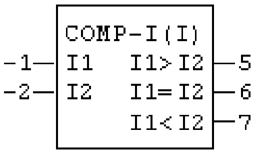


Figure 5-8. PC Element COMP-I

Entry and Presentation Format on Engineering Stations

When entering integers, you can use a minus sign and digits. Two integers can be separated by spaces. There can be no spaces between the minus sign and the digits or between adjacent digits.

Examples of integers are:

1	4096
-4711	-2

Advant Controller 100 Series Internal Format

Internally, integers are stored as follows:

I	2bytes, 2’s complement
IL	4bytes, 2’s complement

5.2.3 Real Numbers

The data type Real (R) is used primarily when processing continuous (analog) signals and also when very large or very small numbers must be handled.

Real numbers are sometimes also called floating point numbers. This refers to the internal representation and is not discussed here.

The range of Real data is approximately:

$-9.2*10^{18}$ to $-2.7*10^{-20}$ / 0.0 / $5.4*10^{-20}$ to $9.2*10^{18}$

Numbers between $-2.7*10^{-20}$ and $5.4*10^{-20}$ are given the value 0.0.

Most elements for arithmetic and data handling and elements for control and filter function work with real numbers.

Figure 5-9 shows a PI regulator in which the outputs O, DEV and all inputs except TI, RINT, and BAL are of type R.

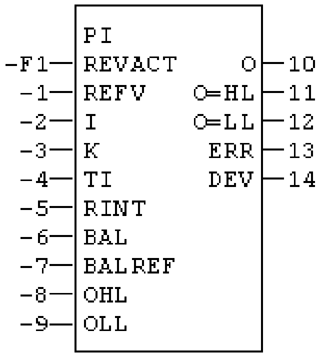


Figure 5-9. PC Element PI

Entry and Presentation Format on Engineering Stations

Digits, ‘+’, ‘-’, ‘.’ (decimal point), and ‘E’ are permitted in a real number.
A real number can be entered in the following forms:

- Number with or without a decimal point,
  for example, 25770 12.459 -12.459
- In E (exponential) format with or without a decimal point, for example,

number 15750 can then be entered as	15.750E3 or 1.575E4
number 1.235 can be entered as	1235E-3
number 0.001 can be entered as	1E-3

An E must always be immediately preceded by a digit. E must also be immediately followed by a digit, or if the exponent is negative, by the minus sign.

If a number contains a decimal point, the decimal point must be either preceded or followed by a digit.
Example: 0.25 must be entered as 0.25 or as .25.

Spaces are not allowed in real numbers.

When generating PC diagrams, values that are larger than 10^{6} or smaller than 10^{-6} are printed in exponential format.

Advant Controller 100 Series Internal Format

Internally, reals are stored as 4 bytes floating point number according to IEEE format.

5.2.4 Time

The following two data types are available to represent time:

T	for times up to 24 hours	(0 ms to 23h 59m 59.999s).
TR	when times longer than 24 hours are required	(0 ms to $9.2*10^{18}$ s).

The data type T is used, for example, for delay elements and oscillators.

The data type TR (Time Real) is primarily used for control and filter functions.

For the delay element TON (Figure 5-10), the data type T is used for the Time Delay input (TD) and the Time Elapsed output (TE).

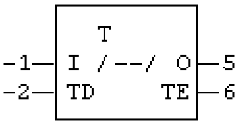


Figure 5-10. TON PC Element

For the integrator INT in Figure 5-11, the type TR is used for the integration time, input TI.

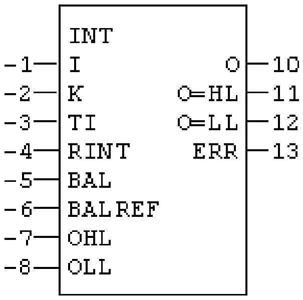


Figure 5-11. PC Element INT

Entry and Presentation Format on Engineering Stations

A time value can be entered in the following format:

Milliseconds	Given for the data type T as a positive integer followed by the unit designation ms. Example: 10 ms, 500 ms. For the data type TR, parts of a millisecond, for example, 10.999 ms, can be entered. This is, however, not meaningful since this precision cannot be maintained in calculations.
Seconds	Given as a real number which may be followed by the unit designations. The unit s is the default. That is, if no unit is given, s is assumed. For type TR, exponential form (E-form) is also permissible. Example: 7.38, 7.38s, 5.2E2, 5.2E2s E must always be directly followed by an exponent digit.
Minutes or Hours	Specified as a real number with unit designation m or h. For the data type TR, exponential format can be used. Example: 25.3m, 18.25h, 1.25E2h

Colon format	The colon sign (:) is used to separate hours, minutes, and seconds. Parts of a second are given with a decimal point. Example: 6:25:30.2 = 6h25m30.2s 25:30.2 = 25m30.2s A colon must be preceded by a hour or minute digit and followed by a minute or second digit. For both data types: - A decimal point must be preceded by a digit and followed by a digit. - A colon may not occur in times specified with a unit (h, m, s, or ms).

Advant Controller 100 Series Internal Format

Internally, time is stored in one of the following formats:

T	4 bytes, positive integer
TR	4 bytes, positive real number.

5.2.5 Group Data

Group data (G) can be used to simplify the handling of a group of associated data of the same data type for example, such as a group of seven integers or a group of 32 real numbers:

- The number of data in a group can be between one and 255.
- Any data type except array can form a group.

A group data unit is specified for an element in the same way as single data units.

Group data can be used, for example, to give input data to the PC element FUNG-1V, a function generator, see Figure 5-12.

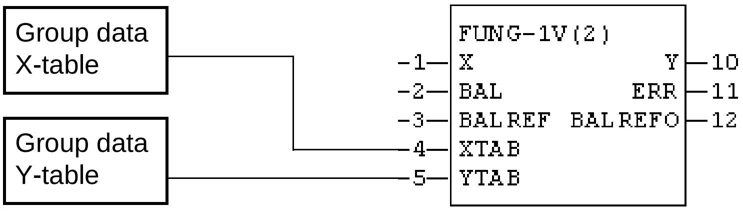


Figure 5-12. PC Element FUNG-1V

An input signal X gives an output signal Y, which follows a specific function. The function generator gives a partial linear curve, by linear interpolation between break points.
A X value and an Y value are required for each break point. These values are stored as group data and are connected to the XTAB and YTAB inputs, respectively.

The size of a group of data can be determined by the PC element itself or by a call parameter. In the example in Figure 5-12, the number is specified with “(2)”.

Entry and Presentation Format on Engineering Stations

No special entry format exists for group data.
To form group data, the PC element REG-G must be used, see Figure 5-13.

This element compresses single data units at the inputs I1, I2, and so on, to a group data unit at the output O. This output can then be connected to a group data input, for example, XTAB or YTAB in Figure 5-12.

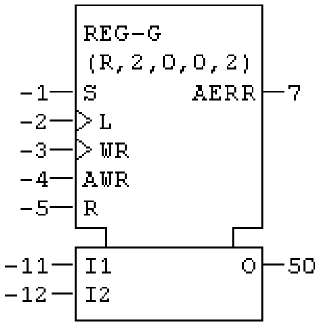


Figure 5-13. PC Element REG-G

Advant Controller 100 Series Internal Format

Internally, group data is stored in the following form:

- 2 bytes that specify the number of data in the group
- A number of bytes according to the space requirements for the type of data in the group padding to an even number of bytes.

5.3 Variables, Constants and Parameters

5.3.1 Inputs and Outputs of PC Elements

Input data to a PC element may be one of the following:

- An output variable from another PC element:
  - Within the same executable unit
  - Within another executable unit.
- A readable variable of a DB element
  - Representing modules or channels
  - Representing common data.
- A constant.

Output variables of PC elements may be connected to one of the following:

- An input variable of another PC element
  - Within the same executable unit
  - Within another executable unit.
- A writable variable of a DB element
  - Representing modules or channels
  - Representing common data.

Storage space in the local data area for variables and constants is automatically reserved when target code is generating.

5.3.2 Common Variables

Common variables are the DB elements of data type:

• Boolean Data	DAT(B)
• Integer Data	DAT(I), DAT(II)
• Integer Long Data	DAT(IL)
• Real Data	DAT(R)

These variables are common for all executable units, and depending on the controller type, for DataSet communication, for local MMI, and for remote communication using MVB.

Refer to Section 4.8.2, Configuration of DAT/PARDAT/DAT_/MDAT Elements for a description of the DAT DB elements.

5.3.3 Constants

A constant is a local variable within a PC program which normally has a fixed value during execution.

A constant may, for example, be used to specify the delay time for a delay element, see Figure 5-14. The constant is specified with D=*value*. In the example, the delay time is 10s.

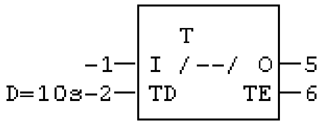


Figure 5-14. Constant at PC Element TON

For reasons of compatibility to MasterPiece 200/1 and Advant Controller 400 Series program constants can also be specified by CD=, MD=, or MCD= instead of D=.

Constants of Boolean type can also be entered as shown in Figure 5-15.

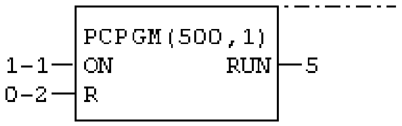


Figure 5-15. Example of Boolean Constants

Each user-defined constant is stored in an unique location within the local data area of the program. A constant may be common to several PC elements, for example, as a common value for several time delays or counters.

5.3.4 Function Parameters

Function parameters are used when the value at a terminal is to be constant for a particular application. Function parameters are fixed by the different PC elements. A function parameter cannot be converted into a normal PC element terminal.

Refer to Section 5.4.1.6, Function Parameters for a description of function parameters.

5.3.5 Common Parameters

Common parameters are the DB elements of type param:

• Boolean Param	PARDAT(B), PARDATS(NB) (subordinate of PARDAT(NB))
• Integer Param	PARDAT(I), PARDAT(II), and PARDATS(NI) (subordinate of PARDAT(NI))
• Integer Long Param	PARDAT(IL)
• Real Param	PARDAT(R)

They can be provided with user-defined initial values.

These parameters are common for all executable unit, and in case of AC 110, 160. and derivatives, for the local HMI (MMI). Refer to Section 4.8.2, Configuration of DAT/PARDAT/DAT_/MDAT Elements for a description of the PARDAT DB elements.

5.3.6 Symbolic Naming of Variables and Constants

Variables and constants can be given symbolic names.

A symbolic name has the following properties:

- Must be unique within a PC program.
- Must begin with a letter.
- Can be up to 20 characters long.
- Can contain letters, digits, and underscore '_'.
  If the name is enclosed in parentheses, special characters can be used. The parentheses are considered part of the name, and the name is printed with them.
- Can consist of uppercase and lowercase letters.

The system does not differentiate between lowercase and uppercase letters so, for example, the name "TID1" is considered the same as "tID1".

Naming Variables

An output variable can be identified by the item designation of its terminal. The output variable can also be given a symbolic name. This name is shown in the PC diagram and can be used as an alternative when connecting elements (see Figure 5-16).

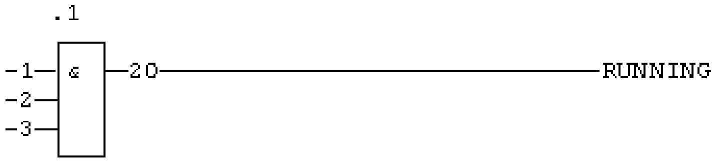


Figure 5-16. Symbolic Name of Output Variable

Naming Constants

Constants can be given symbolic names. See an example of the use of a symbolic name for a constant in Figure 5-17 below.

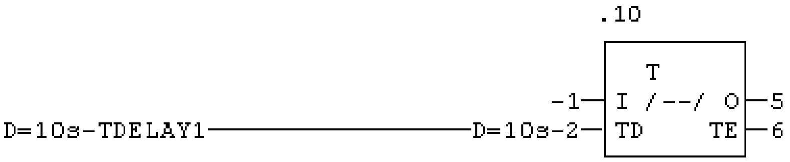


Figure 5-17. Symbolic Name of a Constant

5.4 PC Elements

See Section 5.1.2, PC Elements in General for a brief overview of the PC elements. For a detailed description of the PC elements see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.1 Common Properties of PC Elements

5.4.1.1 Representation

The general graphic symbol for a PC element is shown in Figure 5-18.

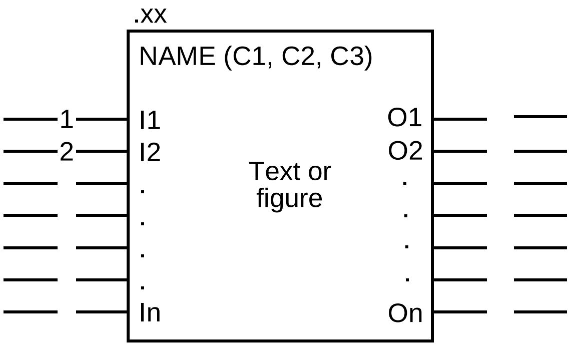


Figure 5-18. PC Element Symbol

Outside and to the upper left of the symbol, you can see the last part of the item designation. Within the symbol, at the top, the call name (see Section 5.4.1.3, Call Names) and the call parameters (see Section 5.4.1.5, Call Parameters) of the element are usually given.
Two deviations from this standard representation are possible:

- The element may contain a figure that illustrates its function (see Figure 5-19).
- The symbol is, to the extent possible, drawn in accordance with available standard symbols, and call name and parameters are then omitted (see Figure 5-20).

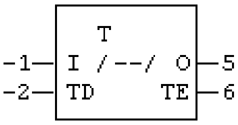


Figure 5-19. Example of Element with Function Symbol

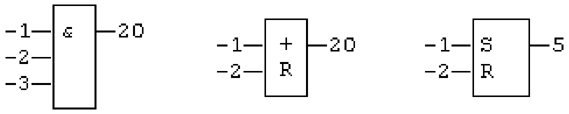


Figure 5-20. Example of Element with Standard Symbol

5.4.1.2 Item Designation

Every PC element is identified by an item designation. These item designations consists of the string “PC” and one or more numbers separated by a point. The numbers consist of one, two, or three digits. The item designation has a hierarchical structure. The item designation of a PC element is unique within every node.

Examples of item designations:

- PC1
- PC1.23.4.234
- PC23.4

Figure 5-21 gives an example of an item designation system for identification of PC elements for the control of a pump installation.

The system has a hierarchic structure similar to the division of a document into chapters and sections. In the example, the PC element in the subsection for pump control has the designation PC1.1.2.1. The item designations are used in program entry and in the PC diagram. They are also utilized as search references when testing and tracing faults in the program.

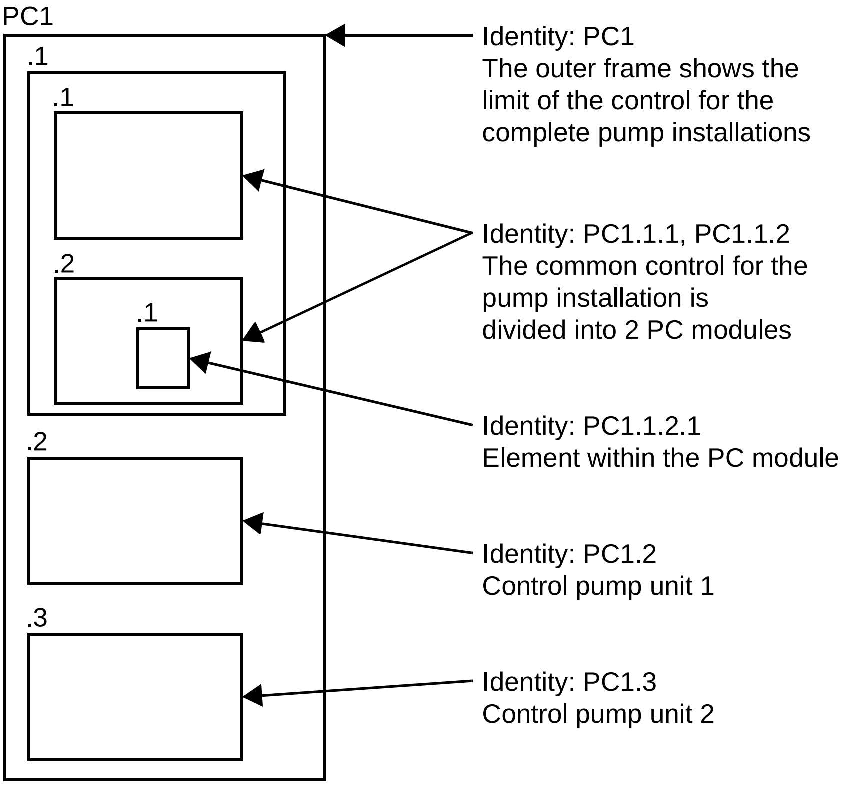

PC1
.1
.1
.2
.1
.2
.3
Identity: PC1
The outer frame shows the
limit of the control for the
complete pump installations
Identity: PC1.1.1, PC1.1.2
The common control for the
pump installation is
divided into 2 PC modules
Identity: PC1.1.2.1
Element within the PC module
Identity: PC1.2
Control pump unit 1
Identity: PC1.3
Control pump unit 2

Figure 5-21. Example of the Item Designation System

5.4.1.3 Call Names

Call Name in General

A PC element’s type is specified by its call name and possibly one or more call parameters for element size, data type, and so on.

The call name is unique and is designed as a mnemonic abbreviation of the full element name. The name can consist of a maximum of eight characters.

Examples:

- COUNT Counter
- TRIGG Triggering element.

Auxiliary Term

The call name can also contain an extra term, for example, SR-D, REG-G, or MUX-I.
The auxiliary term indicates that the element has a special function or, as is the case for MUX-I, specifies the data type that is used for the address inputs. SR-D indicates that the memory element SR has a data input (D input).

REG-G is a register element for group data and is used with, for example, function generators.

The conversion element CONV-IB is used for conversion from Integer to Boolean.

The following designations are used as auxiliary terms in call names:

A	And function (in OR-A elements)
B	Boolean
D	Data (SR-D)
G	Group data
I	Integer
M	Memory function
N	1-of-N addressing (selector)
O	Or function (in AND-O elements)
P	Number of Poles (1P for 1 pole filter, 2P for 2 pole filter)
R	Real numbers
S	Synchronization (in PULSE-S)
T	Time
V	Variable (1V and 2V denotes number of variables for a function generator.)

Combined auxiliary designations are sometimes used, for example, IB, as mentioned above; MN, which denotes 1-of-N addressing with memory function, and so on.

5.4.1.4 PC Instance Names

The Function Chart Builder can assign a name to a PC element instance, a type circuit instantiated in the PC section, or a circuit located in the PC section.

The scope of an instance name is the PC program or—in case of type circuits and circuits—the PC section, in which it is created. All signal names and PC instance names within the scope must be unique.

The length of instance name is 1 to 72 characters.

The PC element instances have no default names.

Since Function Chart Builder 4.7 the PC instance name together with terminal number or terminal name can also be used when connecting PC terminals. For example, you can use `buddy:20` instead of `PC.1.1.2.4:20`.

5.4.1.5 Call Parameters

Many PC elements are flexible with respect to size and data type. Others may have a fixed main function but different input and output configurations.

This information regarding size, data type, and so on, is specified with call parameters that you enter with the engineering station. It is stored in the PC program together with other information regarding the element.

Size Parameters

A size parameter may, for example, specify the number of inputs, groups, or outputs. It may also specify the size of an internal storage area, for example, the number of positions in a shift register.

The calling sequence, AND(C1) where C1 denotes the number of inputs, is specified in the element data sheet for the AND function. The parameter must be an integer in the range 2 to 19, that is, the smallest number of inputs is 2. A 3-input AND-gate is consequently called by AND(3).

A size parameter is used, for example, with the switch elements SW and SW-C.
A size parameter may also specify the size of an internal storage area.

An example of this is shift register SHIFT, where the parameter C2 specifies the number of places in the shift register. In the queue register FIFO, the parameter C2 specifies the number of parallel queues and the parameter C3 specifies the number of places in each queue.

The size of the internal storage area does not influence the representation of the symbol in the PC diagram unless it also affects the number of inputs and/or the number of outputs.

Data Type Parameters

For elements that handle different types of data, the type used in a particular instance of the element is determined with a data type parameter.

The data type is specified in accordance with Section 5.2, Data Types.

A shift register for integers with 16 places is, for example, called with SHIFT(I,16).

The conversion element CONV is used to convert data of one type into another. The call parameters C1 and C2 determine the date type at the input and at the output, respectively. To convert an integer long (IL) into a real number (R), this element is called with CONV(IL,R).

For most elements, the chosen data type is given inside the symbol.

Variant Determining Call Parameters

Certain elements may have different variants of the same main function. The variant used in a particular case is determined with a call parameter given in the manner specified for the element in question.

One example of this is the conversion elements CONV-IB and CONV-BI which convert Integer to Boolean and vice versa. The parameter C2 specifies the code that is assumed at the outputs (CONV-IB) and the inputs (CONV-BI):

- C2 = 1 BC code
- C2 = 3 BCD code
- C2 = 3 1-of-N code
- C2 = 4 Gray-code.

Call Parameters for Structure Elements

For the structure elements with execution control, the following is given with call parameters:

- Cyclicity in ms
- The order of execution among several executable units having the same cyclicity with respect to initial/power-down behavior.

Refer to Section 5.4.3, Structure PC Elements.

5.4.1.6 Function Parameters

Function parameters are used when the value at a terminal is to be constant for a particular application.

The input REVACT in Figure 5-22 for the element PI is an example of such a terminal. The function parameter REVACT (Reverse Action) determines if the output signal is to increase or decrease with increasing actual value at input I.

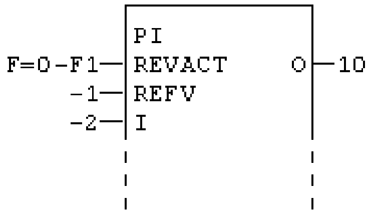


Figure 5-22. Function Parameter of PC Element PI

Function parameters are identified with a number preceded by an F, in this case by F1.

Function parameters are not modifiable by means of Advant Controller 100 Series test functions; they may only be changed within on-line programming.

It depends on the PC element which inputs are function parameters. You cannot set or change the kind of the inputs.

The value is specified in the PC diagram as follows:

F = *value*

The format for the data type of the function parameter determines the entry and representation format of the value.

5.4.1.7 Terminals

A PC element has a number of input and output terminals for connection to other PC elements, DB elements, or constants. Input terminals are always on the left-hand side of the symbol while output terminals are drawn on the right-hand side.

Terminal Numbers

The terminals are identified with numbers in accordance with the element definition.
The terminal number is specified outside the symbol frame, see Figure 5-23.

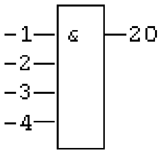


Figure 5-23. Example of Terminal Numbers of an AND Element

An output normally has a terminal number which starts with a multiple of five, for example, the number 20.

If an element has groups containing both inputs and outputs, like the switch element, for example, each such group is normally assigned its own 10-series.

The terminal numbering is fixed for each element and you cannot change it.

Terminal Names

A terminal has an explanatory symbolic name. The name is written inside the symbol frame adjacent to each terminal. Figure 5-24 gives an examples of this.

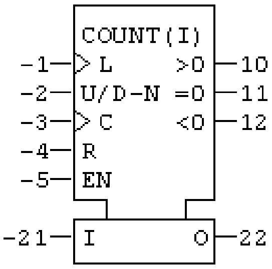


Figure 5-24. Terminal Names of PC Element COUNT

A data input to an element normally has the name I and an output name of O. If several data inputs exist, they are often numbered and given names such as I1,I2, and so on. The outputs are given the names O1, O2, and so on.

The terminal names are fixed for each element and you cannot change them. They are given and explained in each element data sheet.

Terminal Type

In the element data sheet, the data type of each terminal is stated. It may be given, for example, as IB for an Input of type Boolean, II for an Input of type Integer, and so on.

Examples of designations that may occur in the data sheet are given below.

Table 5-1. Designations in Data Sheet

For Inputs	For Outputs
IB	OB
II	OI
IIL	OIL
IR	OR
IT	OT
ITR	OTR
IGRC1	OGRC1

IGRC1 designates an input for group data. This group data contains C1 real numbers. OGRC1 designates an output for group data with C1 real numbers. C1 refers to a call parameter.

Dynamic Inputs and Inverted Signals

Figure 5-25 shows the symbol for a dynamic input. A dynamic input is active only when the input signal changes from 0 to 1.
A static input, on the other hand, is active while the input signal is 1.

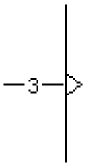


Figure 5-25. Symbol for Dynamic Input

An inversion of a logical input signal is always drawn outside the symbol, see Figure 5-26.

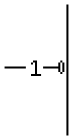


Figure 5-26. Symbol for Inverted Input Signal

5.4.1.8 Special Representation

The basic form for an element symbol is a rectangle with fixed or variable size. The delay element TON may serve as an example of a symbol with a fixed size.

The AND function is an example where the symbol size depends on the number of inputs.

An element may be composed of a common (superior) control part and a data section. Examples of such elements are counters and registers. Both the control and the data part of these elements may be of variable size.

The switch elements SW and SW-C, for example, have a size which varies with the number of contact groups (channels).

The symbol layout, terminal numbers, and so on, are fixed for each element and are shown in the element data sheets.

5.4.2 Basic PC Elements

Basic PC elements (or function elements) are:

• Logic elements	AND, OR, INV, SR, SR-D, …
• Time elements	TIMER, OSC-B, …
• Data handling elements	FIFO, REG-G, SW, MUX-I, …
• Arithmetic elements	ABS, ADD, COUNT, COMP-I, …
• Function generators	FUNG1-V, …
• Feedback and filter algorithms	INT, PI, FILT-1P, …
• Special elements	SYSL, SYSTIME, …

In case of AC 110, 160, and derivatives also

Master Fieldbus Communication MFB-IN, MFB-OUT, …

5.4.3 Structure PC Elements

Structure PC elements are:

• PC program module header	PCPGM
• Control module headers	CONTRM, MASTER, SEQ

• Function module header	FUNCM
• Block module header	BLOCK
• Step module header	STEP
• Slave module header	SLAVEM.

5.4.3.1 Purpose of Structuring

The structure elements are superior PC elements with two purposes:

- By using structure PC elements, the PC program can be arranged into hierarchically ordered function sections, PC modules, in accordance with the structure of the process.
- The structure PC elements (except FUNCM) control, on a module-by-module basis, the execution of the PC elements in the PC program.

Each PC module consists of at least one structure PC element, the module header. In the PC diagram, the module header is placed in the upper left-hand corner of the module frame, see Figure 5-27.

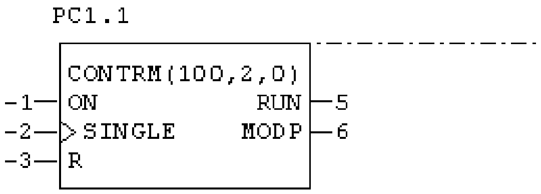


Figure 5-27. Module Header in a PC Module

5.4.3.2 Structuring a PC Program into PC Modules

To obtain an easily handled control system, the PC program can be divided into functional sections. The partitioning of the program is usually adapted to the functional structure of the plant, process, or machine that is to be controlled. The need may arise to turn on or off different modules. Different parts of the process may have different timing requirements, and so it may be necessary to execute different program modules with different cycle times.

A well-structured program easily gives overview documentation and promotes efficient program development, commissioning, and maintenance.

The system permits subdivision into nine levels.

There are three rules for program subdivision into function sections:

- The subdivision of the program should parallel the division of the plant, process, or machine.
- If the execution of any part(s) of the program must be different from other parts, the program has to be further subdivided to allow for this.
- Try to develop standardized program modules which may be reused in other programs.

5.4.3.3 Hierarchy of the Module Headers

In a PC module, either other modules or basic PC elements can be included. Basic elements and module headers cannot be mixed within a module.The exception to this rule is the PC block (BLOCK), which can be mixed with basic elements.

A module can be composed in one of the ways shown in Figure 5-28 The subordinate modules in Case 2 can then in turn be composed according to one of the two cases.

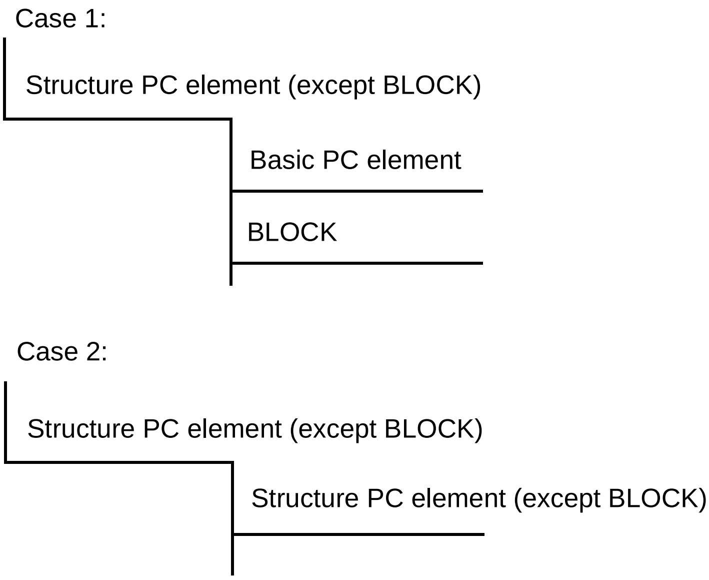

Case 1:
Structure PC element (except BLOCK)
Basic PC element
BLOCK
Case 2:
Structure PC element (except BLOCK)
Structure PC element (except BLOCK)

Figure 5-28. The Two Cases for Composing PC Modules

5.4.3.4 PC Program Module PCPGM

The programming part or PC part of the user application program is built in a hierarchical structure.

On the top level of this structure, there is the PC element PCPGM (**PC P**ro**G**ra**M**).

On the engineering station, up to 99 PCPGMs can be created for one node.
Advant Controller 100 Series, however, is able to store and run only one PCPGM.
Advant Controller 100 Series systems with CPU redundancy, such as AC 160, are able to store and run only one PCPGM in each CPU module.

The PCPGM forms an executable unit to control the execution of the other executable units (the control modules), and if applicable, to control the dataset transfer of common data (see Section 4.10, Advant Fieldbus 100 Network Configurations).

A PC program must contain one or more control modules to have a meaningful function.
A PC program is a unifying frame for a number of execution units. You can also turn execution of all execution units contained in the program on and off from the program control header.

For a detailed description of the PCPGM element, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.3.5 Function Modules FUNCM

A function module is created by introducing a function module header FUNCM into the program hierarchy.

A function module has its own identity in the program hierarchy.

The PC elements in the function module are given identities one level down in the hierarchy.

A function module has no execution conditions, so it must be part of an execution unit, or alternatively, contain execution units.

A function module may itself contain function modules but, at some level of the hierarchy, there must exist a control module to determine execution for the PC elements of the program. Refer to the chapters on these modules.

For a detailed description of the FUNCM element, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.3.6 Control Modules CONTRM, MASTER, SEQ

A control module generates, in addition to the PC program module, an executable unit on the Advant Controller 100 Series system.

The control module CONTRM is the most general execution unit in the Advant Controller system. The control module MASTER works in the same way as the control module CONTRM, but the subordinated PC elements in the PC diagram are arranged in another structure
(see Section 5.4.3.7, Master-Slave Combination MASTER/SLAVEM).

The control module SEQ generates an execution unit for sequence control (see Section 5.4.3.8, Sequence-Step Modules SEQ/STEP).

The basic PC elements in a control module are always executed in the documentation order and with the cyclicity specified for the module.

The control module header is the element that controls the execution of the module in accordance with the conditions in effect.

A control module in a PC program

- May be included together with other control modules in a function module.
- May contain a number of function modules.
- May contain both PC blocks and PC elements.

For a detailed description of the CONTRM, MASTER, and SEQ elements, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.3.7 Master-Slave Combination MASTER/SLAVEM

Master-slave is used when you want to combine modules from different parts of the program hierarchy into a single execution unit, keeping the functional structure unchanged. A master-slave combination is comparable to a control module divided into several function modules.

The MASTER header corresponds to the CONTRM header and controls the execution of those PC elements that are included in the slave modules which belong to the MASTER header.

The slave module header is given by the structure PC element SLAVEM.

Several master-slave groups may be included in a PC program. The individual slave modules may be located at different levels in the hierarchy. They may also be included in function modules together with control modules or other slave modules.

The master-slave group is executed as a unit. The slave modules belonging to one master are all executed after one another in the order of documentation.

Master-slave in the hierarchy:

- A master must always be located directly under the PC program PCPGM.
- A slave module can be included, with other slave modules, in a function module.
- Basic PC elements and PC blocks can be included in a slave.
- A slave module can contain function modules.

For a detailed description of the MASTER and SLAVEM elements, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.3.8 Sequence-Step Modules SEQ/STEP

Sequential control is the control of a process or machine performing a predetermined sequence of steps.

A sequence is described in a manner which is in accordance with the standard IEC 1131-3. For a detailed description, see Section 5.5.3, Application with Sequences.

The sequence consists of a superior sequence header SEQ with the common functions for start, initializing, and so on, and a number of step modules, one for each step in the sequence. Each step module has its own step module header STEP whose function is determined by a number of control signals.

The sequence of process events is divided into a number of steps. Only one step can be active at one time. While a certain step is active, the conditions for step change are checked and a change is performed if the conditions are fulfilled.

It may sometimes be advantageous to regard a process as a sequence, even if it is not sequential itself. The result may be a simplified and well-structured control program.

A sequence is composed of a number of step modules.

A step module can be subdivided into function modules. In any case, all PC elements in the step module must be included in one of the function modules.

A sequence is an independent execution unit and so may not be included in any execution unit other than a PC program. A sequence can be included in a function module together with, for example, other sequences, control modules, or slave modules.

A sequence is executed cyclically. Even if no step is active, the sequence header (SEQ) is executed once each cycle to check its control inputs. Each time the sequence header is executed and the sequence is active, the active step module is also executed.

The conditions for step change are also examined.

The step is normally executed in a predetermined order. First, the step header is executed, then the PC elements of the active step are executed, and finally, the system checks the condition for step change.

When the condition for step change from step N to N + 1 is satisfied, the RUN output of step N is reset to 0 and the elements in the step are executed once in normal mode.

Following this, during the same cycle, the RUN output of step N+1 is set to 1 and the PC elements of this step are executed for the first time. During this first execution of a step, the condition for step change to step N + 2 is ignored. This implies that each step is activated during at least one cycle, that is, only one step change is performed per cycle.

For a detailed description of the SEQ and STEP elements, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.4.3.9 PC Block BLOCK

The PC block BLOCK is a possibility to switch the execution of part of a control module on or off.

PC blocks are used if there are functions within an execution unit that need only be performed on particular occasions.

The processor load of the system can also be reduced by controlling a group of PC elements so that they are only executed when necessary.

The PC block is identified (item designation) in the same way as the other elements in the PC module.

Blocks can be included in function, control, or slave modules together with PC elements.
A block can only contain PC elements.

When a block is switched off (ON = 0), the following occurs:

- Data calculated in the block remain unchanged.
- Output of data to the data base and to modules outside the block continues as long as the execution unit to which the block belongs is executed.

When an execution unit is reset, the block modules in the execution unit are also reset, whether they are on or off.

For a detailed description of the BLOCK element, see *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.

5.5 PC Programming

PC programming is performed as follows:

- Suitable PC elements are first selected from an element library and the corresponding symbols are entered in a draft diagram. One element library is selected to suit the process function.
- The elements are then interconnected to provide the required function.

The complete diagram is designated the PC diagram and resembles, in principle, a traditional logic and control diagram.

Some basics follow that apply to a PC program:

- Identification system of all PC elements
- Rules for connection of elements.

5.5.1 Identification of PC Elements

As previously described, the identification system for PC elements in an Advant Controller system is built up in a hierarchic form, similar to the division of a book into chapters, sections, and paragraphs. Figure 5-30 shows an example of the identification principle. It may be easier to understand this principle if the program module division is drawn in structure tree form as shown in Figure 5-29.

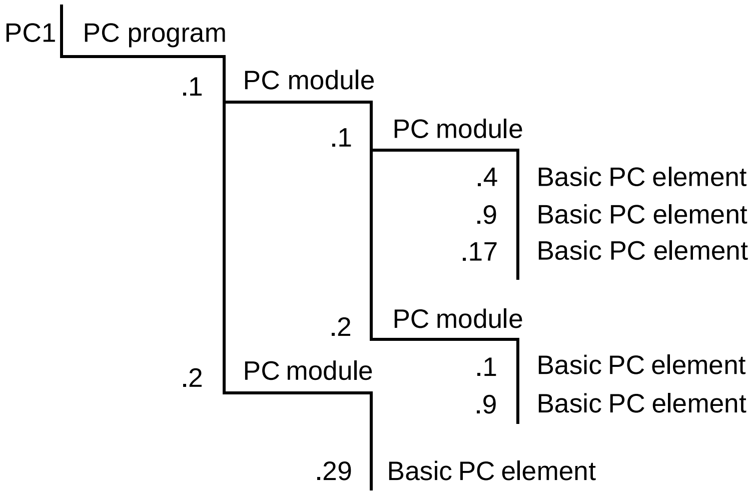


Figure 5-29. Example of a Tree Form

PC Program

The PC program PCPGM has an item designation between 1 and 99.
On an AC 110, 70, 55, or derivative, only one PCPGM is allowed.
On systems with CPU redundancy, such as AC 160, only one PCPGM is allowed for each CPU module.

PC Modules

Each PC program is divided into one or more PC modules which have item designations PC1.1 to PC1.n in PC program PC1, and PC2.1 to PC2.n, in PC program PC2, and so on. On the next lower level, the item designations are PC1.1.1, PC1.1.2, and so on. The maximum number of levels in a program, including the program and element levels, is nine. Each level, except the element level (see below), can be further divided into from 1 to 255 new modules program including.

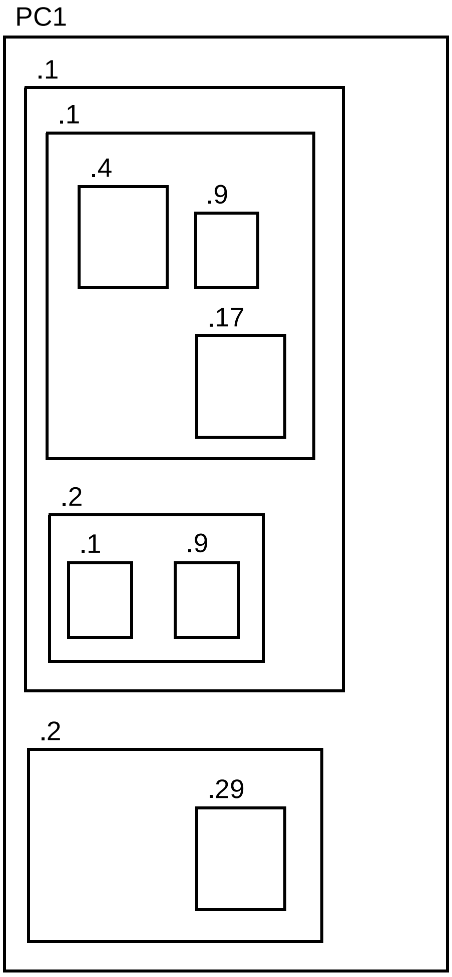


Figure 5-30. Example of a Division into Modules

Basic PC Elements

The elements in a PC module are given item designations ranging from 1 to 255, that is, a PC module can contain a maximum of 255 elements.

The common part of the item designation for a PC module or an element is printed above the upper left-hand corner preceded by a dot (.), except for the highest level (PC programs).

The item designation

- Is used to divide the program into a hierarchical structure.
- Is used to show relations within the program (for example, cross-reference lists).
- Can be used to connect PC elements together or to DB elements when you are creating or modifying application programs.

5.5.2 Connection of Elements

In the PC diagram, a connection between two PC elements is indicated with a line between the corresponding element terminals in the same manner as in a conventional logic diagram.

This connecting line denotes a data exchange between the two elements and is always specified in this way, irrespective of the data type. If the elements are located on different diagram pages, the PC item designation of the element with the output is printed at the connecting line of the element with the input. A cross-reference is also printed for the diagram pages.

Connections to the data base or to constant parameters are specified with a reference at the connecting line for the output or input concerned.

When entering a PC program from a draft PC diagram, the output variable, data base variable, or constant which is to be connected to each input is entered. In other words, the source of each input variable is given.

A symbolic name for an output variable can be defined.

An output variable can be connected to several elements within and between different PC modules without restrictions.

When you are connecting the receiving elements, refer either to the item designation of the output variable, or to its symbolic name, or the PC instance name together with terminal number or terminal name. The names must be previously declared before they can be used as reference.

Feedback

An output variable of an element may be returned to an input of the same element or to an input of an earlier element in the same or in another module.

Such a signal feedback is always delayed one execution cycle.

Inverted Signal

A logical (Boolean) variable can be inverted when it is connected to an input terminal. This is indicated in the PC diagram by a small circle at the corresponding terminal, as shown in Figure 5-31.

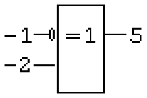


Figure 5-31. Inverted Signal

Inversion of DB Element Connections

If an output variable connected to the data base is to be inverted, an inversion element must be used.

Dynamic Input

A dynamic input is a function of the PC element, so it is connected in the same way as any other input.

Data Type Checking

Data exchange or connection between two element terminals is permissible only if both terminals have the same data type. Upon entry and change, the engineering station checks the respective data type and generate an error message if they do not match.
If it is necessary to connect inputs and outputs with differing data types, for example, in arithmetic expressions, conversion elements must be used.

Interconnection of Outputs

Outputs from two or more elements may not be connected to a single output variable.
The engineering station checks for this condition upon entry and editing and generates an error message.
It is, however, possible to connect to a terminal in the data base from different executable units. This is meaningful only when the executable units are executed alternately.

Connection to Group Data

When you are connecting group data, both data type and the number of data in the group must match between output and input. The data output of the register element, therefore, has the form G*nd* where:

- G denotes group data.
- *n* denotes the number of data items in the group.
- *d* denotes the data type.

A register element for 10 real numbers will get the data type denotation G10R for its data output in the PC list.

Connection to Data Base

When you are connecting to the data base, reference is made to the identity of the data base variable concerned or to its symbolic name. This applies to both input and output variables.

When several element inputs use the same data base variable as input, source reference may be made either to the data base variable itself or to the first user of that variable.

When an output variable is to be connected to more than one data base variable, each connection is automatically assigned its own PUT element. Refer to Section 11.3, Input and Output Phases for a more detailed description.

An output variable connected to the data base may, at the same time, also be connected as an input variable to another PC element.

If, instead, reference in the example above is made to the data base when connecting the input, the new value of the output signal will be available one execution cycle later.

Connection to Constants

A constant is a local variable which normally has a fixed value during execution.

It is denoted D=, CD=, MD=, or MCD= *value*.

The different denotations have the same meaning, but for compatibility reasons, they are limited to MasterPiece 200/1 only.

Boolean constants can also be entered as 0 or 1.

A constant can be common to several terminals.

PC Element Variables

An output variable of a PC element represents data at the output of the element. The PC element COUNT in Figure 5-32 has four output variables, each representing data calculated in the element during execution.

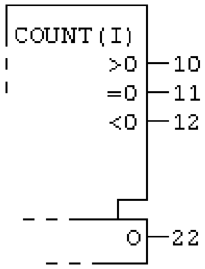


Figure 5-32. Output Variables of PC Element COUNT

An output variable is normally identified with the identity of the element together with the terminal number or terminal name. Since Function Chart Builder 4.7 the PC instance name together with terminal number or terminal name can also be used when connecting PC terminals. For example, you can use `buddy:20` instead of `PC.1.1.2.4:20`.

The value of an output variable is transmitted to the input of an element by connection.

A connection is represented in a PC diagram by a connecting line (see Figure 5-33) or by the item designation of the connected terminal (see Figure 5-34).

For connected PC elements within different executable units, the output terminal is marked with a P for Put. The input terminal is marked with a G for Get, to indicate the internal move elements (see Figure 5-35).

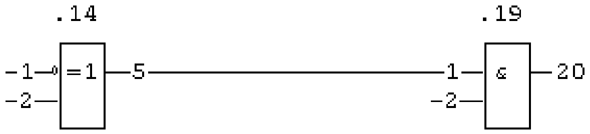


Figure 5-33. PC Element Connection Within the Same Page and Execution Unit

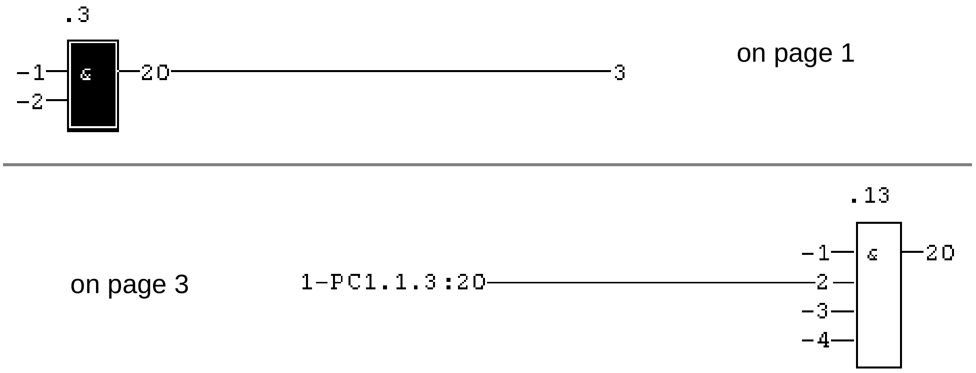


Figure 5-34. PC Element Connection at Different Pages but the Same Execution Unit

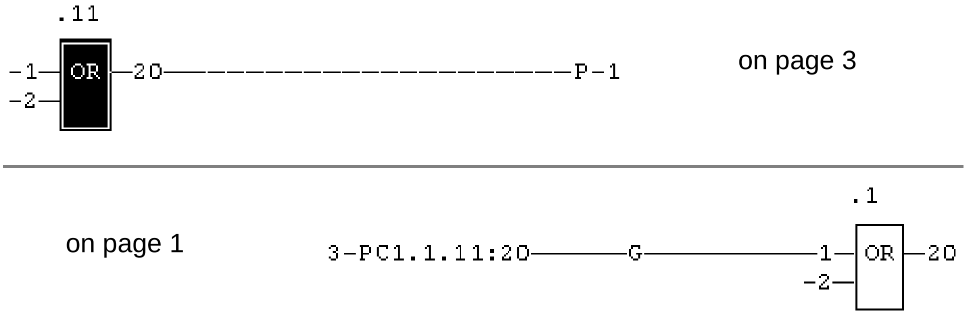


Figure 5-35. PC Element Connection at Different Pages and Different Execution Units

Connections to DB Elements

Data outside the PC program is referenced via DB elements. This reference is made by a simple ‘=’ notation in the PC diagram, see Figure 5-36.

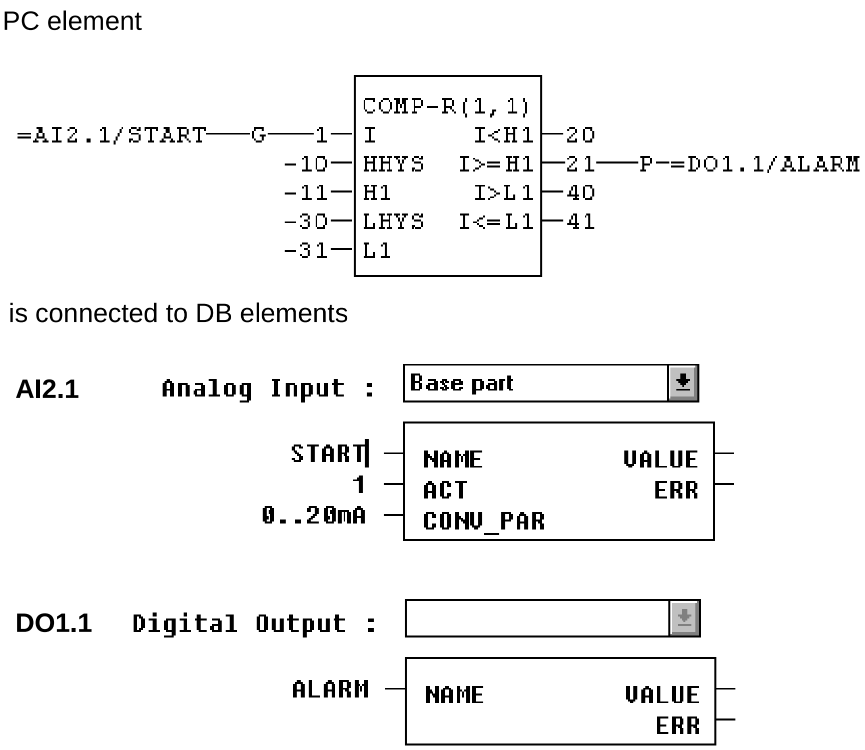


Figure 5-36. Connection to DB Element in a PC Diagram

The = sign denotes a connection to a DB element. Either the name or the item designation can be used for identification. When the output VALUE of the DB element is referenced, it is not necessary to give this name. When referencing other terminals of the DB element, the terminal name must be given preceded by a colon (:).

Examples: =DI3.2 =DI3.2 (:VALUE) =START:ERR.

All connections to DB elements are marked at the output terminal of the PC element with a P for Put and at the input terminal with a G for Get to indicate the internal move elements (see Figure 5-36).

In all documentation, the identity is written independently of the entry format. The diagrams or print-outs show the item designation of the DB element, the element name, and the terminal name or, alternatively, the hardware address (BUS, STATION, POSITION) of the module element and, if existing, the channel number, the element name, and the terminal name.

Example: =DI1.7/SEQUENCE_MAN:VALUE

5.5.3 Application with Sequences

The function of the sequence and step modules is described in the *PC Elements Advant Controller 100 Series Reference Manual*, *PC Elements Advant Controller 70 Reference Manual*, and *PC Elements Advant Controller 55 Reference Manual*.
Construction of straight sequences and sequences with branches and reunions is described below.

5.5.3.1 Function Diagram

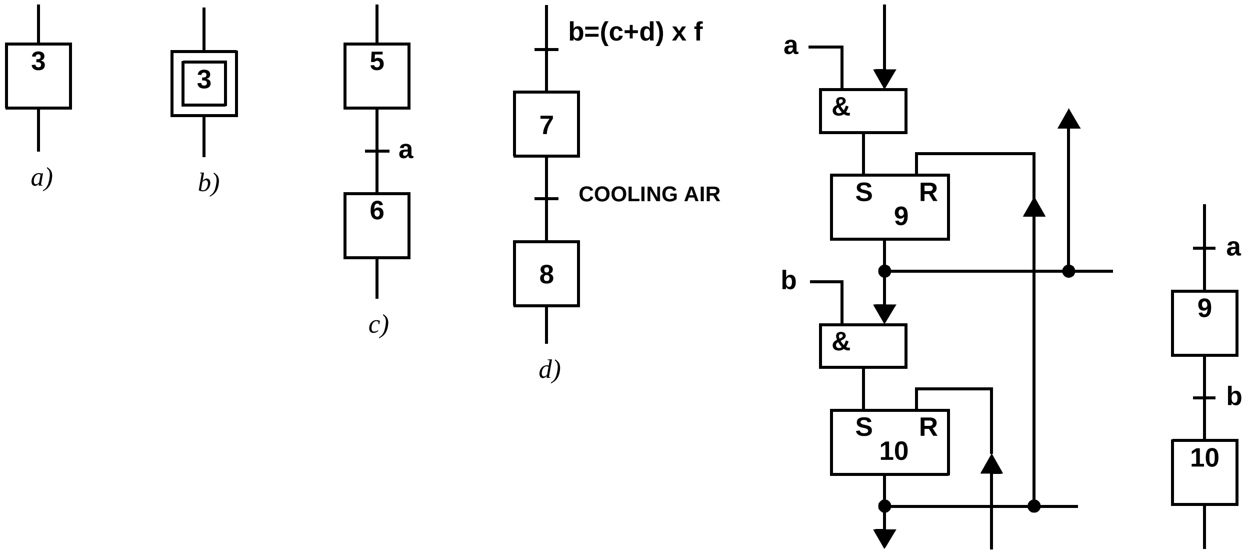
Symbols according to standard IEC 1131-3 were used in the figures in this section.
You should always prepare such a diagram when you are constructing control programs with sequences. This provides the structure and main functions of the control in an easily surveyed document.
According to the IEC standard, each step in the sequence is indicated by a square in which the number of the step is written, see Figure 5-37a.

An initialization step is indicated with a double square as in Figure 5-37b.

The condition for step change (transition condition) is indicated with a line across the connecting line between two steps. In Figure 5-37c the transition condition is that the variable **a** is true. That is, step change is performed if step 5 is active and the variable **a** is true.

There must always be a transition condition (a simple variable or an expression) between two steps. This condition may be stated either in plain language or in Boolean form, as shown in Figure 5-37d. There can only be one condition between two steps.

Figure 5-37e shows a comparison between the function diagram presentation and the corresponding circuit, in logic diagram form.



a)

b)

c)

d)

e) Comparison Between Diagram and Circuit Presentation

Figure 5-37. Examples of Transition Conditions

5.5.3.2 Activities

The activity in each step is written in a separate box connected to the step square as shown in Figure 5-38a. If several activities are to be performed in the same step, each activity is allocated its own box, as shown in Figure 5-38b.

The activity box can be divided into three fields, designated a, b, and c in Figure 5-38c:

a. Type of activity denoted by a letter:

N (or no character)	Continuous order as long as the step is active
L, D	Time-controlled order
S	Activity with memory function, that is, the activity is also to continue after the step has been active (S = stored).

b. Activity including any additional conditions

c. A reference number if several activities occur in the step. This field is left out if no references are needed.

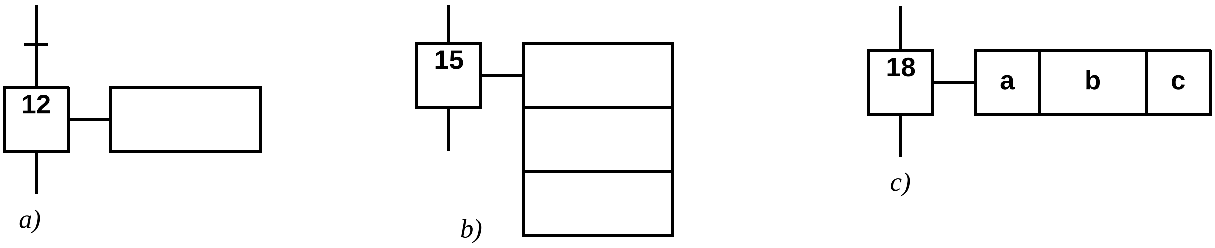


Figure 5-38. Activities

Continuous Order (N)

An N order means that the activity is to continue as long as the step in question remains active. The activity can also be combined with different conditions so that the order is active only when the condition is true.

In Advant Controller 100 Series systems, the RUN output of the step module header is 1 while the step is active.
This output can be used for the N activity. In the example of the car sequence in Appendix A.2, Application Examples, the RUN output may be connected directly to the physical outputs for motor forward/reverse via the data base.

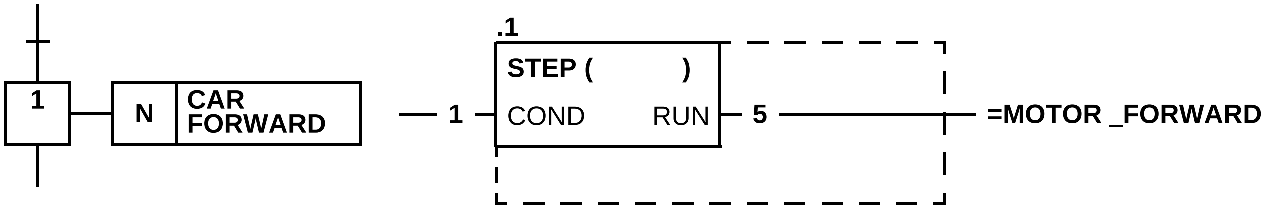


Figure 5-39. Examples of N Order

Any additional conditions, that is, a hatch in the bottom of the car must be closed for the car to run, are implemented by introducing an AND gate in the step module as shown in Figure 5-40.

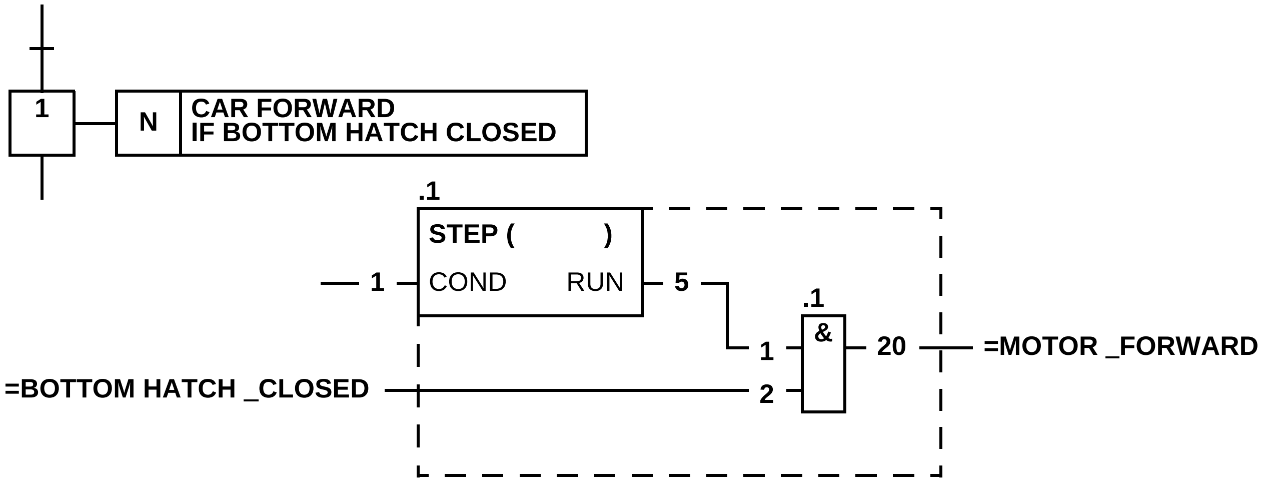


Figure 5-40. Examples of Continuous Order with Condition

Time-controlled Order (D or L)

An activity can be delayed or time-limited. Such time functions are specified with the letter D (delayed) or L (time-limited) and are identified with the number of the step.

Delayed Activity

Figure 5-41a shows a delayed activity: A valve is to be opened 10 seconds after the activation of step 6. Figure 5-41b) shows the Advant Controller 100 Series implementation.

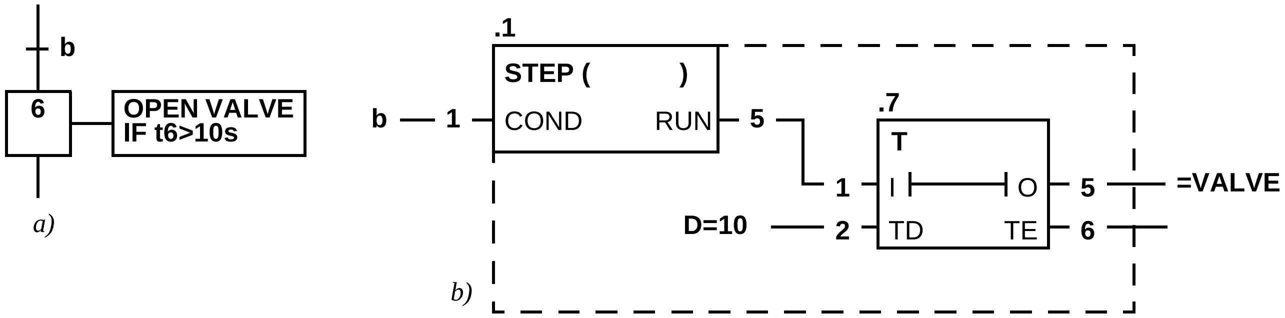


Figure 5-41. Example of Delayed Activity

The internal delay circuit in the step module header can also be used. See Figure 5-42.

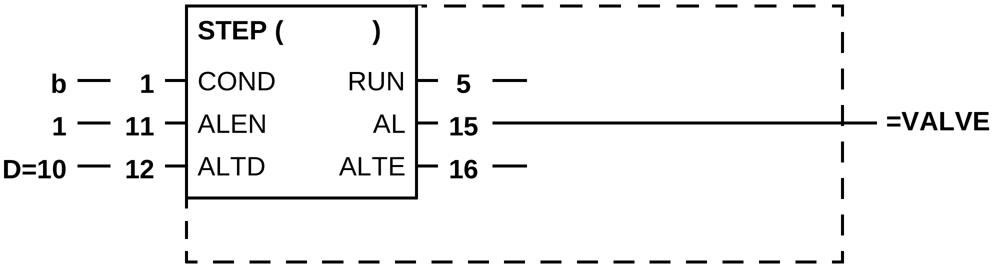


Figure 5-42. Step Module Header for Delayed Activity, Example

Time-limited Activity

A time-limited activity is specified in accordance with Figure 5-43. In the example, the activity “HEATING” is to start when step 14 is activated and is to continue for 25 minutes. It is assumed that the step is active at least as long as the duration of the activity.

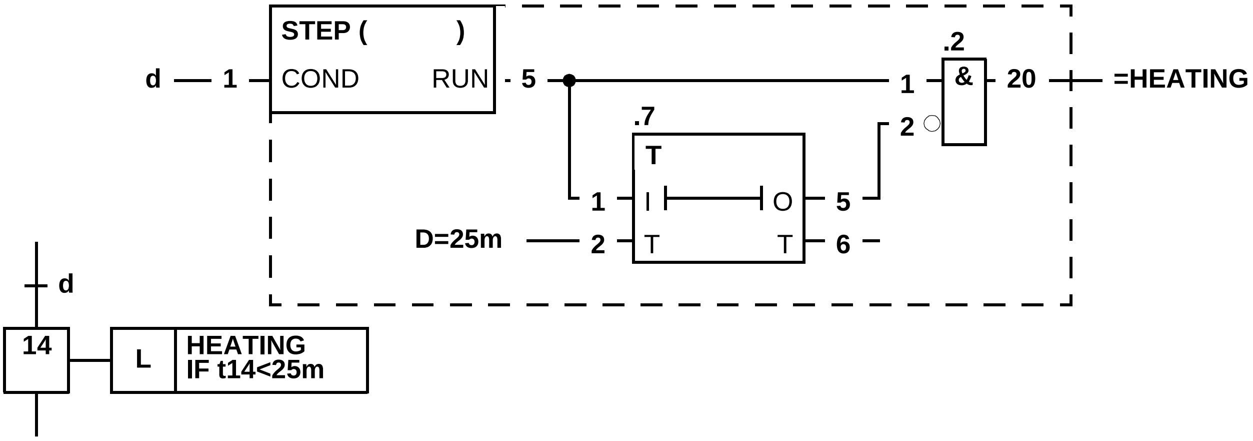


Figure 5-43. Examples of Time-Limited Activity

An example of a time-limited activity that often occurs is a step that is to be active for a specified time. This is shown in Figure 5-44, which also shows an implementation.

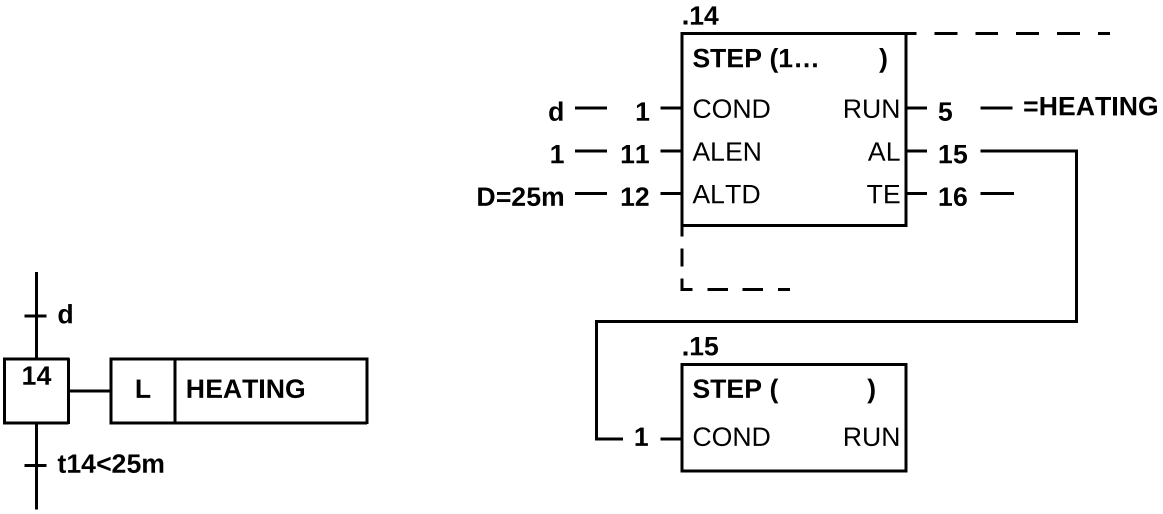


Figure 5-44. Example of Step Being Active for a Certain Time

Activities with Memory Function (S)

A S-activity is an activity that starts in a step and continues even after the transition to the next step. An example is a start-up sequence in which a number of activities are to be started in a specific sequence.

Another example is a specific activity to be started in one step and stopped a number of steps later in the sequence.

It is not necessary to take any special measures to implement an S-activity in an Advant Controller 100 Series system. The execution of a step ceases when it is left. The data calculated in the step remains unchanged until the step is reset or re-executed.

A S-activity can also be implemented by activating a data base variable in one step and then later, in another step, deactivating that variable as shown in Figure 5-45.

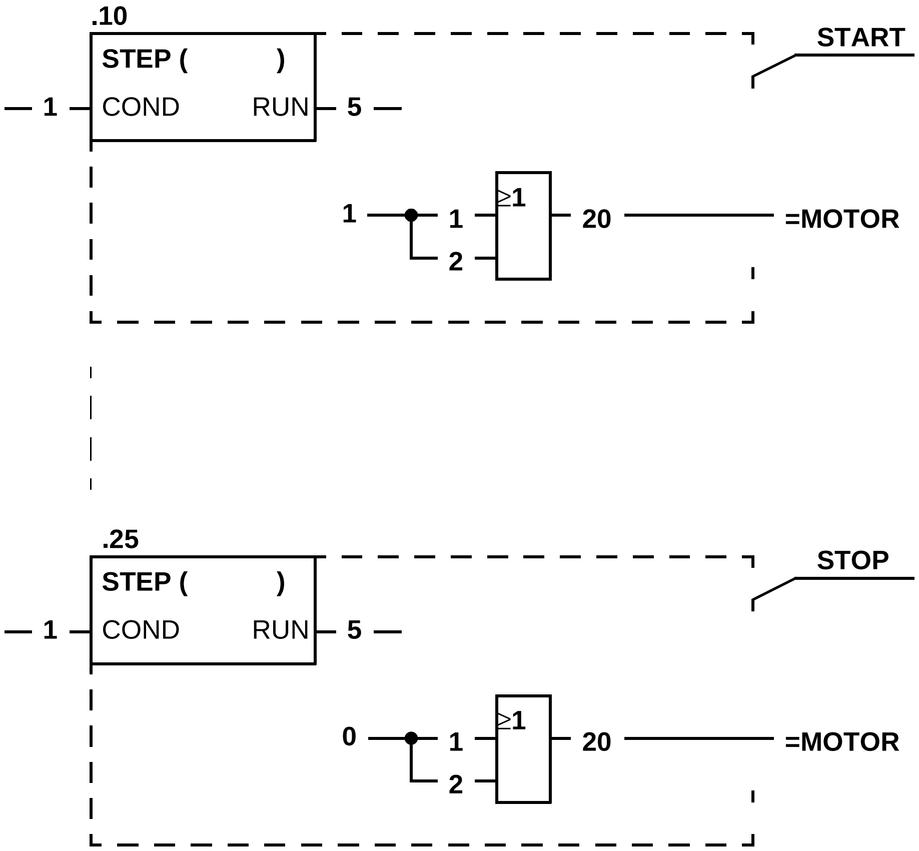


Figure 5-45. Example of Activity with Memory Function

A motor is started in step 10 by setting the data base variable MOTOR to 1. When step 25 is activated, the motor is stopped by setting MOTOR to 0.

Another implementation of this example is shown in Figure 5-46. If you use the programmable reset function of the step header, step 10 is reset when step 25 is exited. This solution is suitable when several activities are to be reset simultaneously.

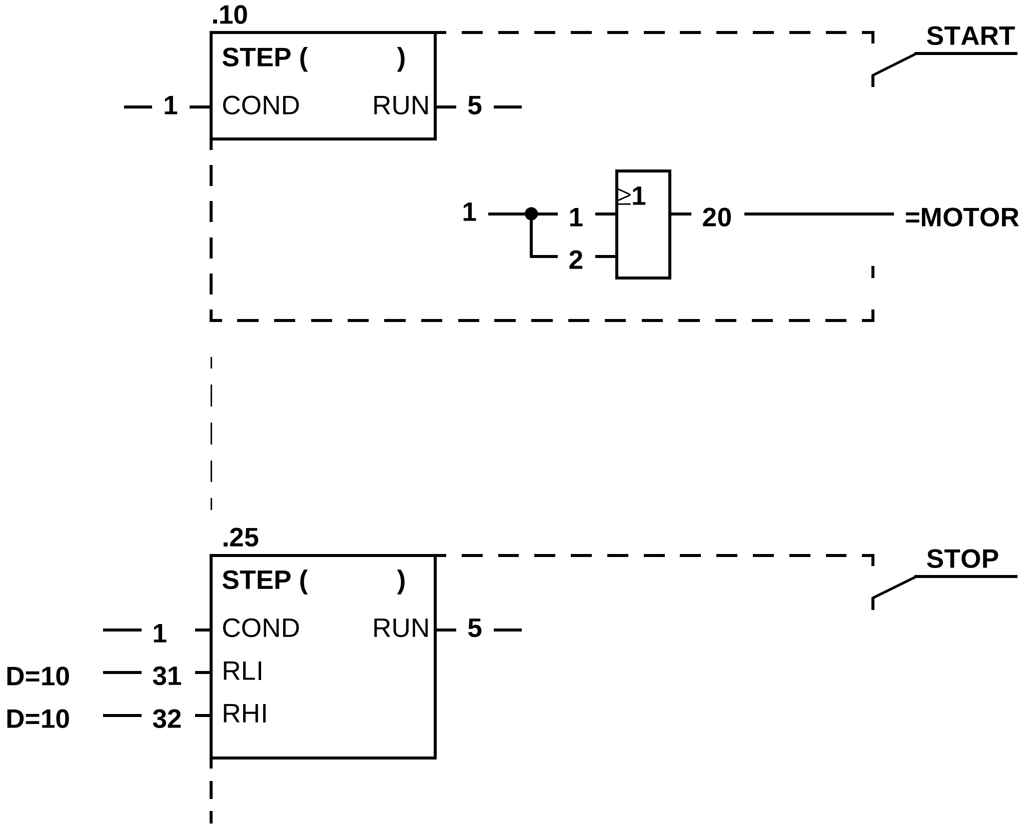


Figure 5-46. Additional Example of Activity with Memory Function

Resetting a sequence implies resetting all steps in the sequence. You must consider this if a data base variable is to be active after the sequence is reset.

5.5.3.3 Single Sequences

A single sequence is made up of a series of steps which are activated one after another.

Start

To start the sequence, activate the START (or STARTC) input. To reset the sequence, activate the R input.

To activate step 1 in Figure 5-47, fulfil the start condition **a** of the step. If the COND input is connected to a constant 1, the step is automatically activated when the sequence is started.

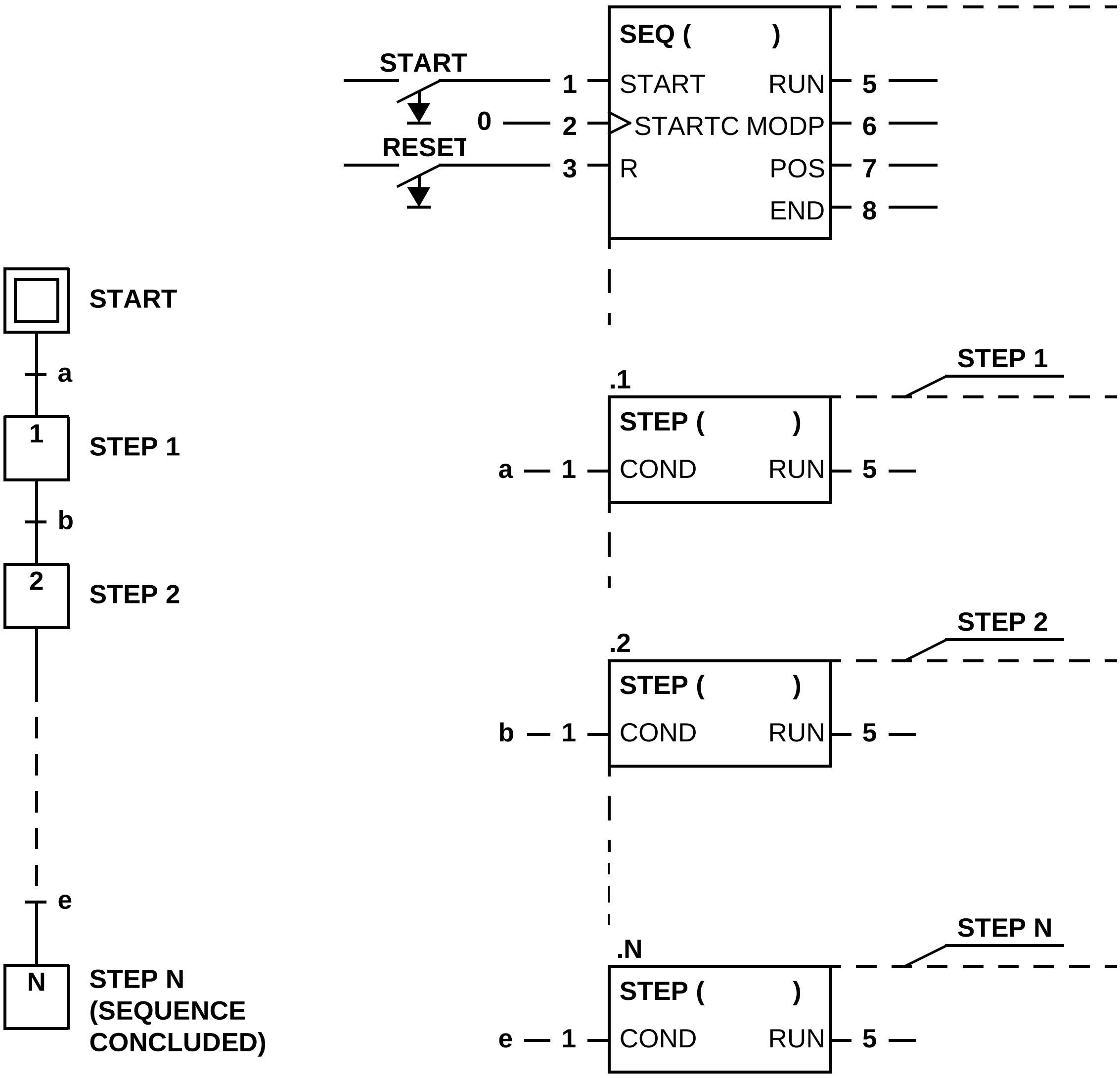


Figure 5-47. Example of Single Sequence

Automatic Resetting

A sequence can be automatically reset when it reaches its last step if, for example, the RUN output of the last step is connected to the R input of the sequence header. The same effect is achieved by connecting the END output of the sequence header to the R input. END is active when the last step of the sequence is active.

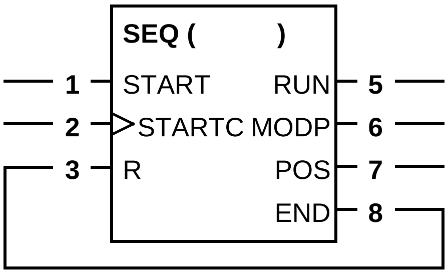


Figure 5-48. Automatic Resetting

Automatic Resetting with Restart

The sequence is automatically restarted if the START input is constantly held at logic 1. In Figure 5-49, the sequence restarts as long as the AUTO switch is closed.

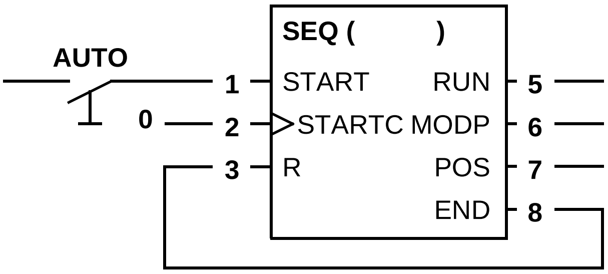


Figure 5-49. Automatic Resetting with Restart

Automatic Resetting Without Restart

To avoid unwanted restarts caused by a start order remaining when the sequence is reset, use the dynamic input STARTC as shown in Figure 5-50.

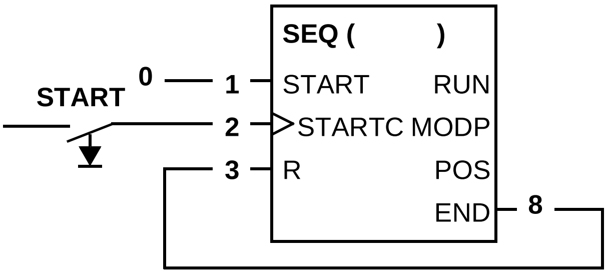


Figure 5-50. Automatic Resetting Without Restart

5.5.3.4 Beginning of Sequence Selection (Divergence)

A sequence selection is a conditional choice between two or more branches. In the example in Figure 5-51, the sequence is to take the left branch if the condition ‘**e**’ is satisfied and the right branch if the condition ‘**f**’ is satisfied. If both conditions apply, the right branch is selected since the jump conditions are examined first.

This type of structure may sometimes also be called divergence.

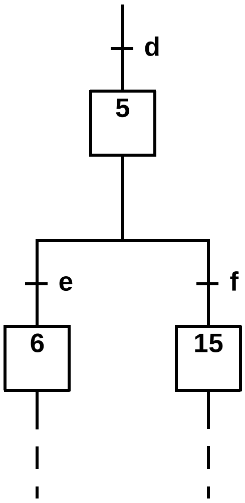


Figure 5-51. Sequence Selection

Figure 5-52 shows an implementation of this branch. You can see the sequence as a single sequence with a side branch from step 5.

For step 5, the call parameter C2 = 1, that is, one jump function. The main branch is laid out in the normal way.

The JCOND1 input of step 5 is connected to the jump condition f and the JPOS1 input is connected to the jump address D=15.

NOTE

Set the COND input of step 15 to 0 to prevent an active step 14 from causing an unintended transition to step 15.

You can specify up to four branches, in addition to step change to the next step, in each step of a sequence.

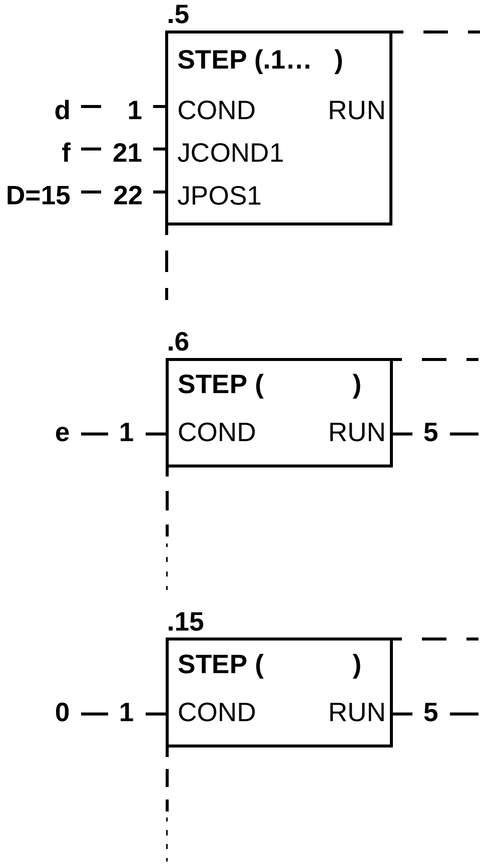


Figure 5-52. Branch in Advant Controller

Figure 5-53 and Figure 5-54 show an example with a suggested implementation.

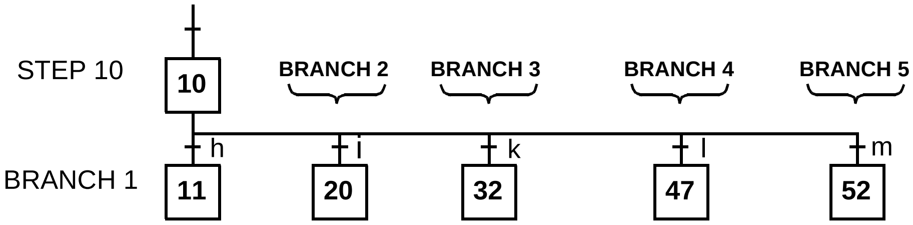


Figure 5-53. Branch

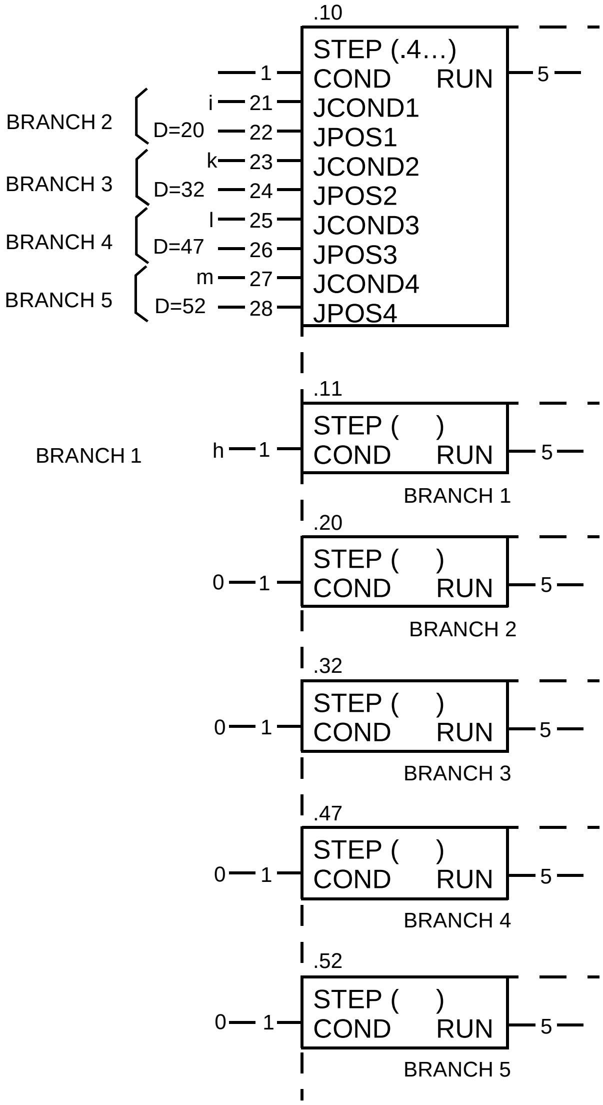


Figure 5-54. Implementation of the Branch in Figure 5-53

The jump addresses in the example are given as operational parameters (D=...). They can also be given as constants (CD=...).

A jump address can be calculated by the program in one of the steps of the sequence or in another program section.

A branch can be controlled by, for example, an operational selector in the control desk.

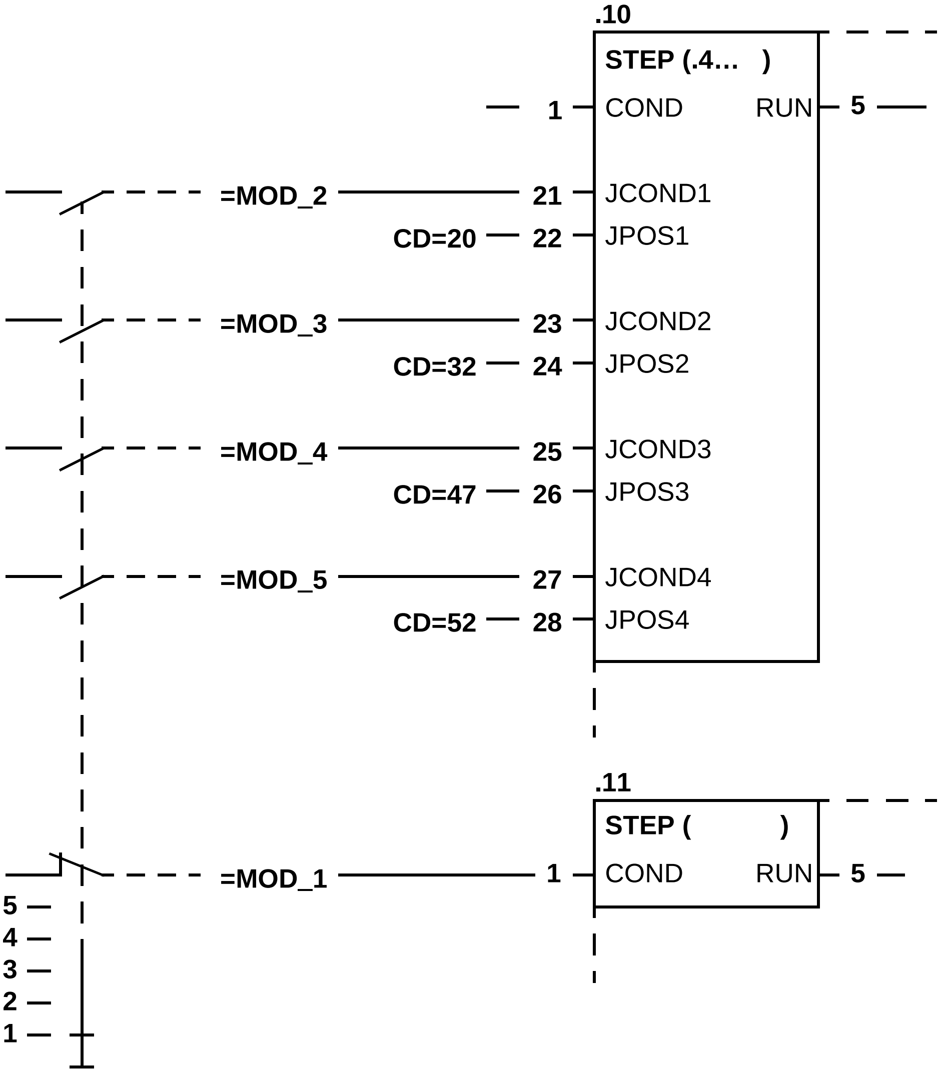
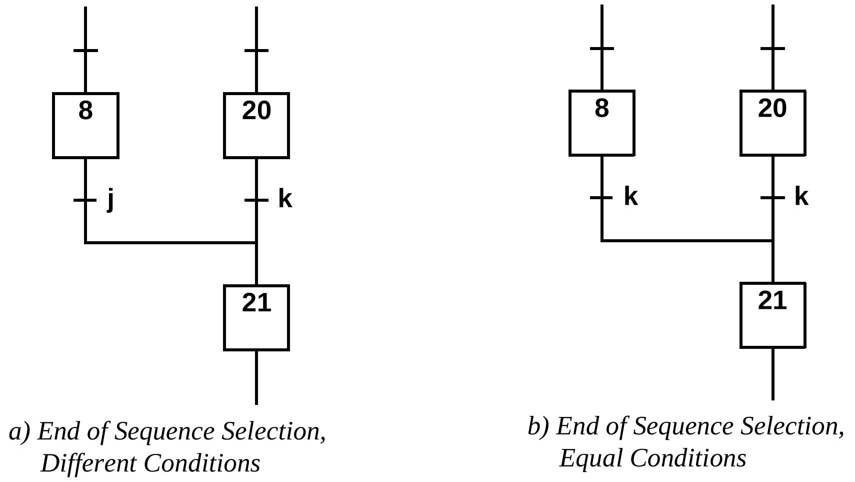


Figure 5-55. Branching Controlled by a Selector

5.5.3.5 End of Sequence Selection (Convergence)

The end of sequence selection of two or more alternative branches is implemented, in principle, in the same way as branching by means of jumps defined by JCOND and JPOS. This end is also often called convergence.



a) End of Sequence Selection,
Different Conditions

b) End of Sequence Selection,
Equal Conditions

Figure 5-56. End of Sequence Selection

Figure 5-56a shows a sequence with two branches converged in step 21. Two different conditions for continuation into the common branch are given in Figure 5-56a. Nothing prevents the use of the same condition, as shown in Figure 5-55b.

The convergence is implemented by giving a jump to step 21 from step 8 with the jump condition j as in Figure 5-57. The COND input of the step after step 8 (step 9) must be set to 0 to prevent an unintended step change from 8 to 9.

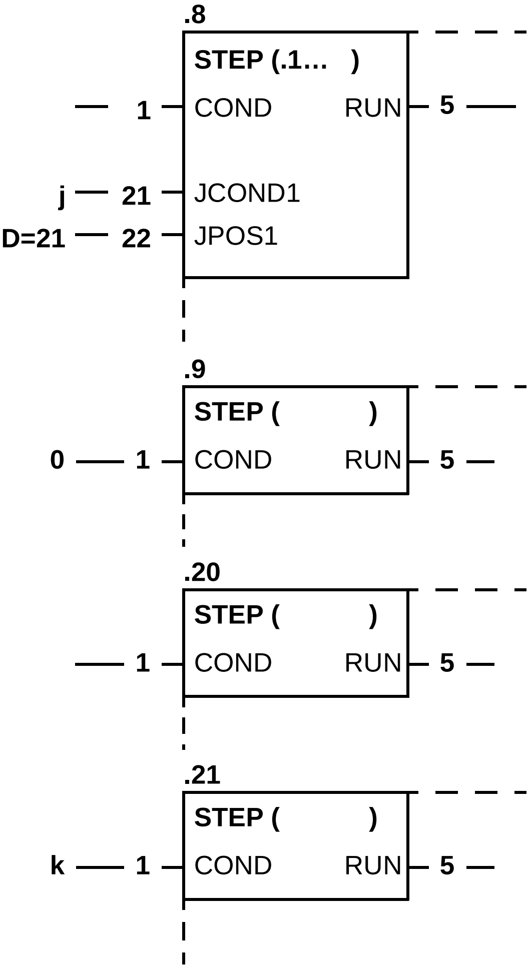
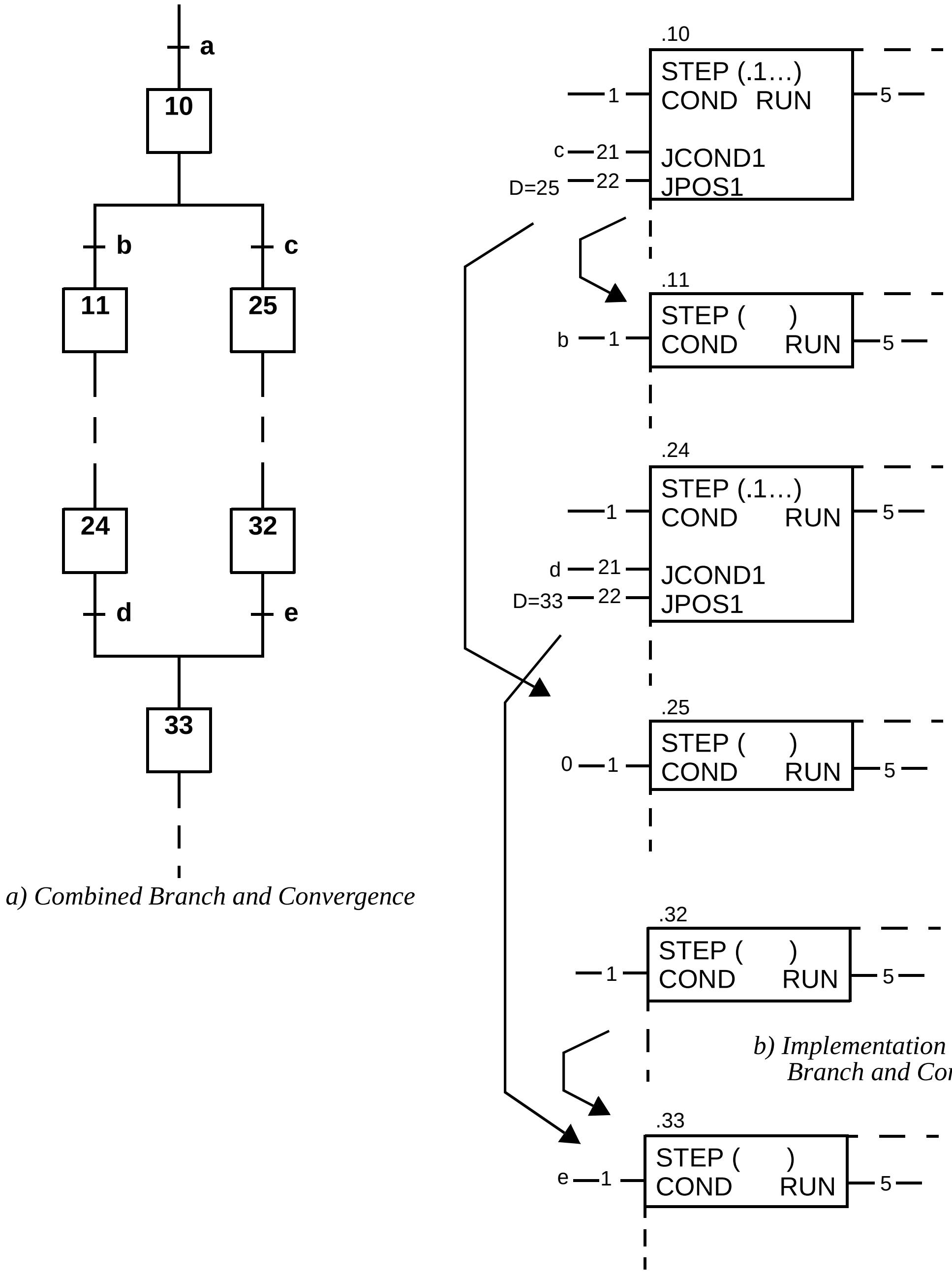


Figure 5-57. Implementation of the Convergence in Figure 5-56a

5.5.3.6 Complete Sequence Selections

A sequence can have alternative branches for parts of the sequence. Figure 5-58a shows an example. The implementation with a combination of a branch and a convergence is shown in Figure 5-58b.



a) Combined Branch and Convergence

b) Implementation of Combined Branch and Convergence

Figure 5-58. Combined Branch and Convergence and Their Implementation

5.5.3.7 Sequence Selection with Several Terminating Branches

In a sequence with several terminating branches as in Figure 5-59, each branch must be finished with a jump to a last step common to all branches. This is necessary to obtain a single last step of the sequence.

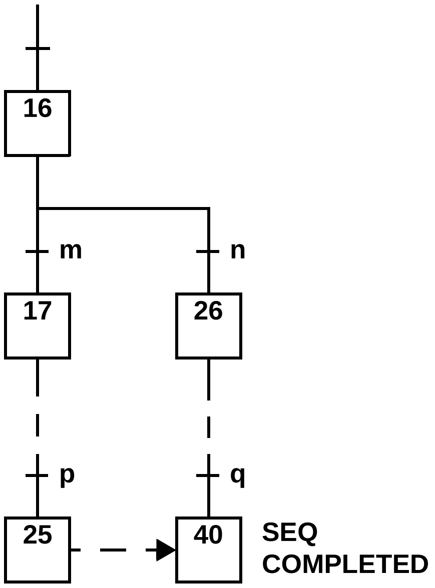


Figure 5-59. Branch with Terminal Branches

The implementation depends upon the layout of the particular sequence. See Figure 5-60 for a suggested solution.

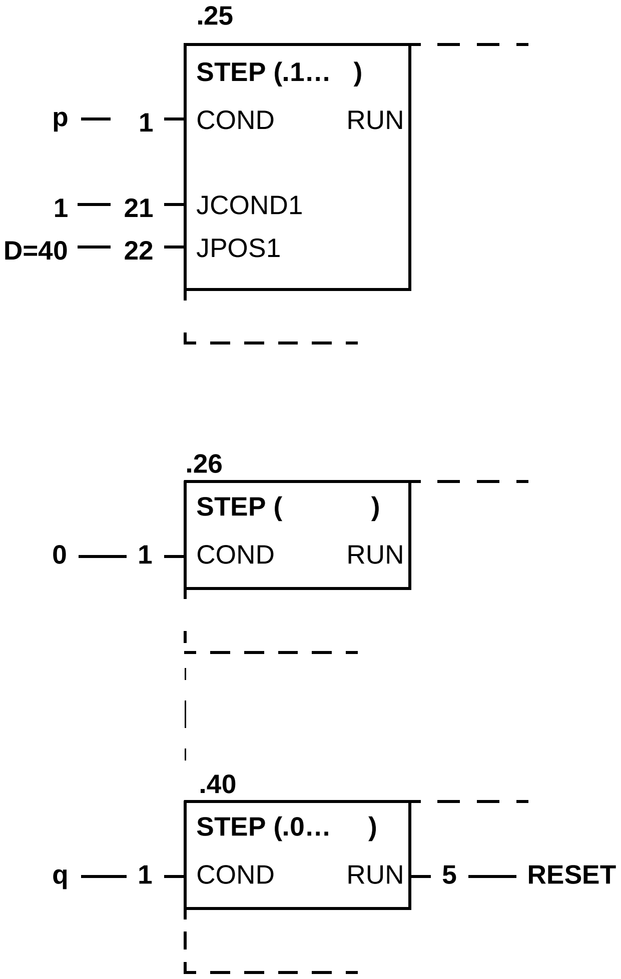
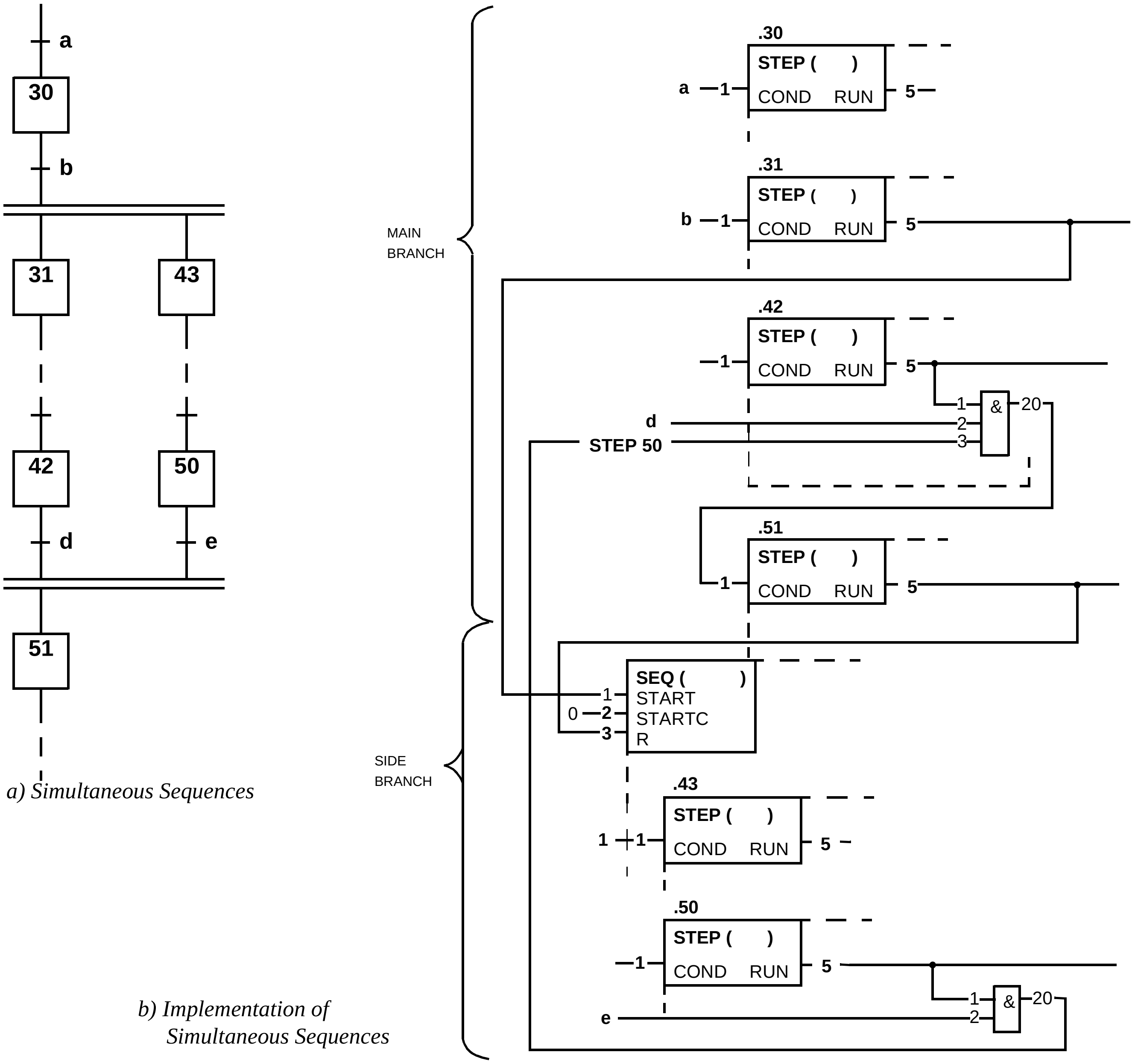


Figure 5-60. Implementation of a Sequence According to Figure 5-59

5.5.3.8 Simultaneous Sequences

If there are to be simultaneously active branches in a sequence, for example, as shown in Figure 5-61a, the side branches must be separate sequences.

Both branches must run to the convergence step before the main sequence proceeds.



a) Simultaneous Sequences

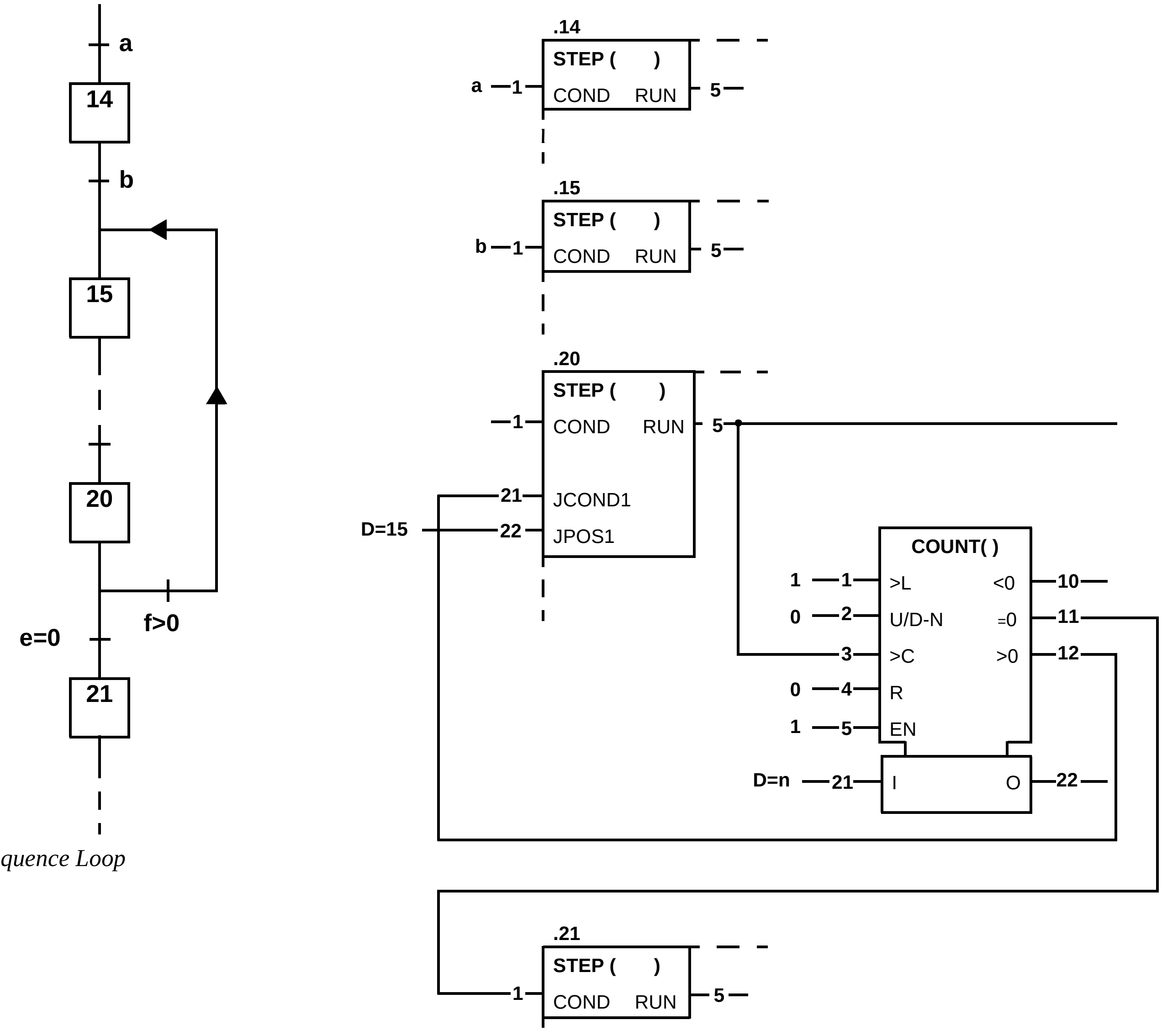
b) Implementation of Simultaneous Sequences

Figure 5-61. Simultaneous Sequences and Their Implementation

5.5.3.9 Sequence with Loop

A sequence can contain a loop that is to be executed a number of times before the sequence continues. The loop in Figure 5-62a is to be executed ‘**n**’ times. A test of the number of times the loop has been executed is performed in step 20. If all ‘**n**’ executions are done, the sequence proceeds to step 21. Otherwise the loop is executed again.

Figure 5-62 shows a solution. The counter in step 20 is preset to ‘**n**’ and is decremented by one each time the step is executed.



a) Sequence Loop

b) Implementation of Sequence Loop

Figure 5-62. Sequence Loop and Its Implementation

5.5.3.10 Use of HOLD

The HOLD function is used if one or more objects controlled by a sequence are to be controlled temporarily in another way.

Suppose in the example of the car sequence in Appendix A.2, Application Examples, you want to be able to stop and reverse the car if an obstacle is encountered. Figure 5-63 shows a solution using the HOLD function. Control of the car is taken over by a module for manual control if the switch in the figure is placed in the HOLD position.

- Normally you cannot manually control activities other than those of the active step without losing synchronization of the sequence since the execution of the step modules is stopped on HOLD.
- An object which can be manually controlled and is started in a sequence step must, in a manual control module, have a latch, that is, in the way shown in Figure 5-63 (MOTOR M).

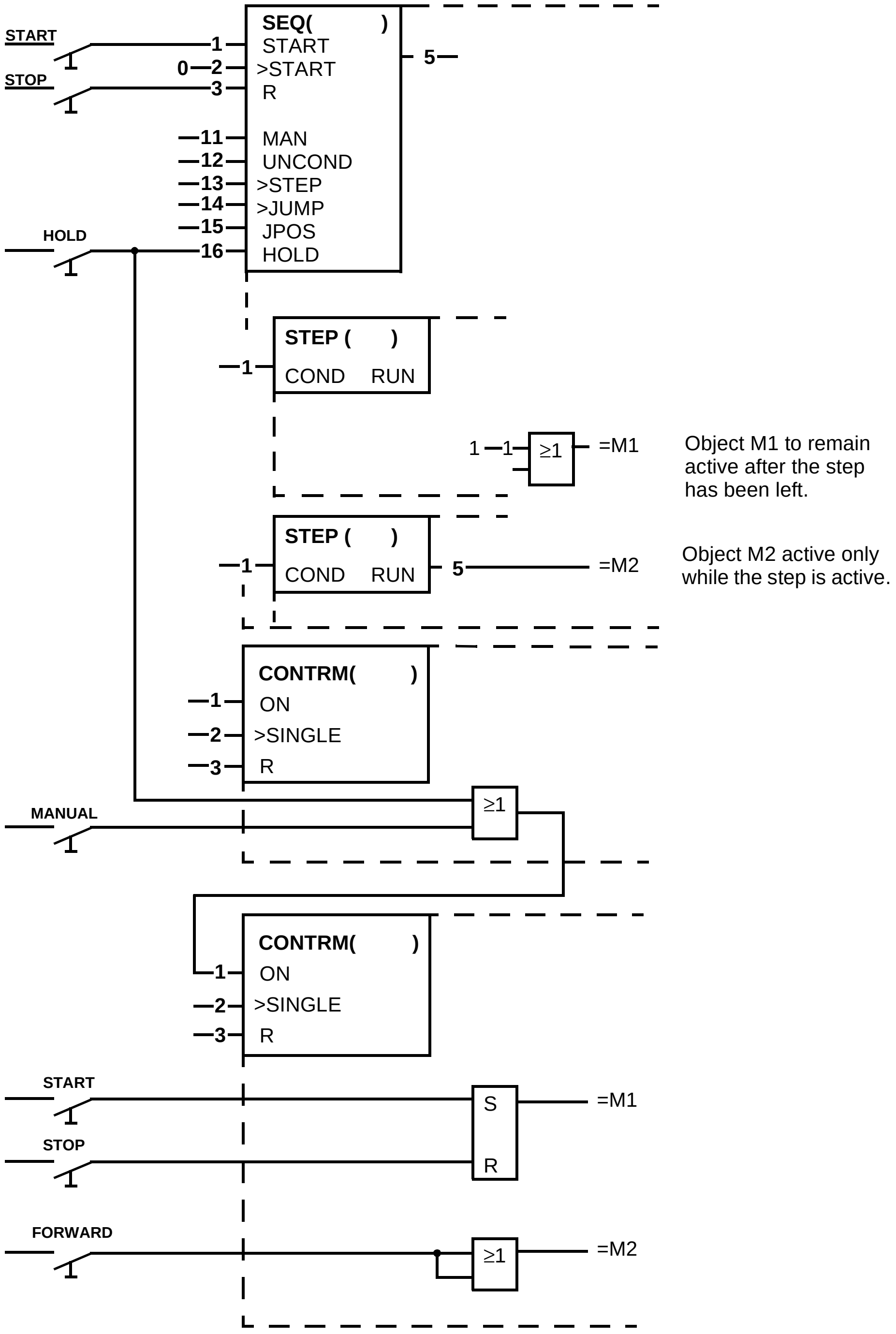


Figure 5-63. Example of Manual Control

5.5.3.11 Dividing a Sequence into Steps

The sequence of process events is divided into a number of steps. Only one step can be active at one time. While a certain step is active, the conditions for step change are checked. The change is performed when the conditions are fulfilled.

Sometimes it may be advantageous to regard a process as a sequence, even if it in itself is not sequential. The result can be a simplified and well-structured control program.

5.5.3.12 Sequence in the Hierarchy

A sequence is composed of a number of step modules. See Figure 5-64.

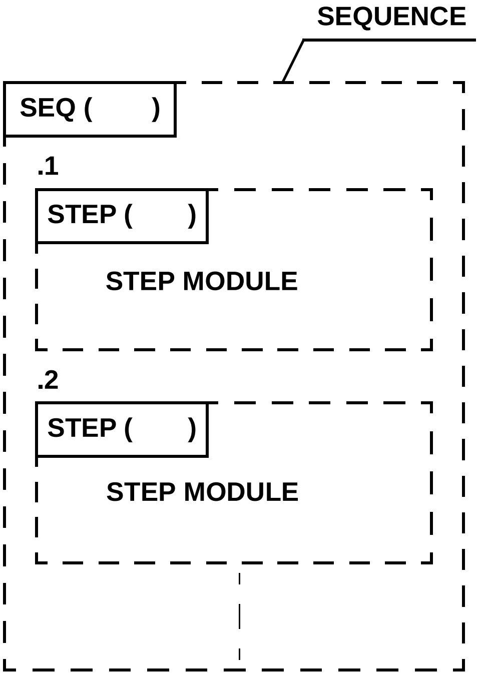


Figure 5-64. Hierarchical Subdivision

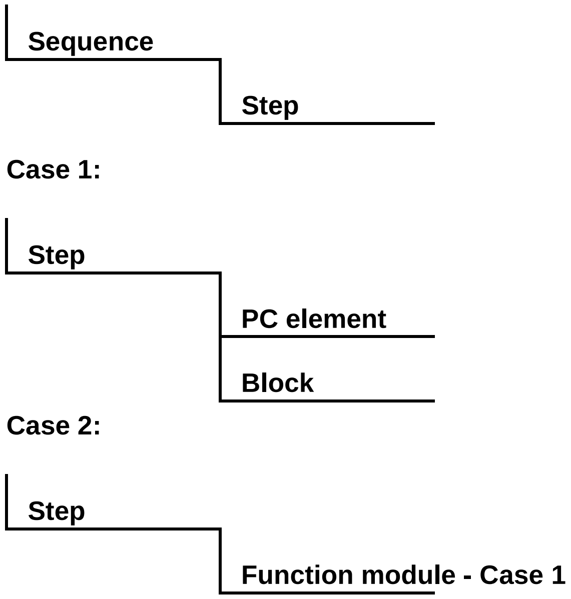


Figure 5-65. Sequence and Step in the Hierarchy

A step module can be subdivided into function modules. In this case, all PC elements in the step module must be included in either of the function modules.

A sequence is an independent execution unit and so cannot be included in any other execution unit but a PC program. A sequence can be included in a function module together with, for example, other sequences, control modules, or slave modules.

5.5.3.13 Function

A sequence is executed cyclically. Even if no step is active, the sequence header (SEQ) is executed once each cycle to check its control inputs. Each time the sequence header is executed and the sequence is active, the step module which is active at the time is also executed.
The condition for step change is also examined. Figure 5-66 shows the principle for this.

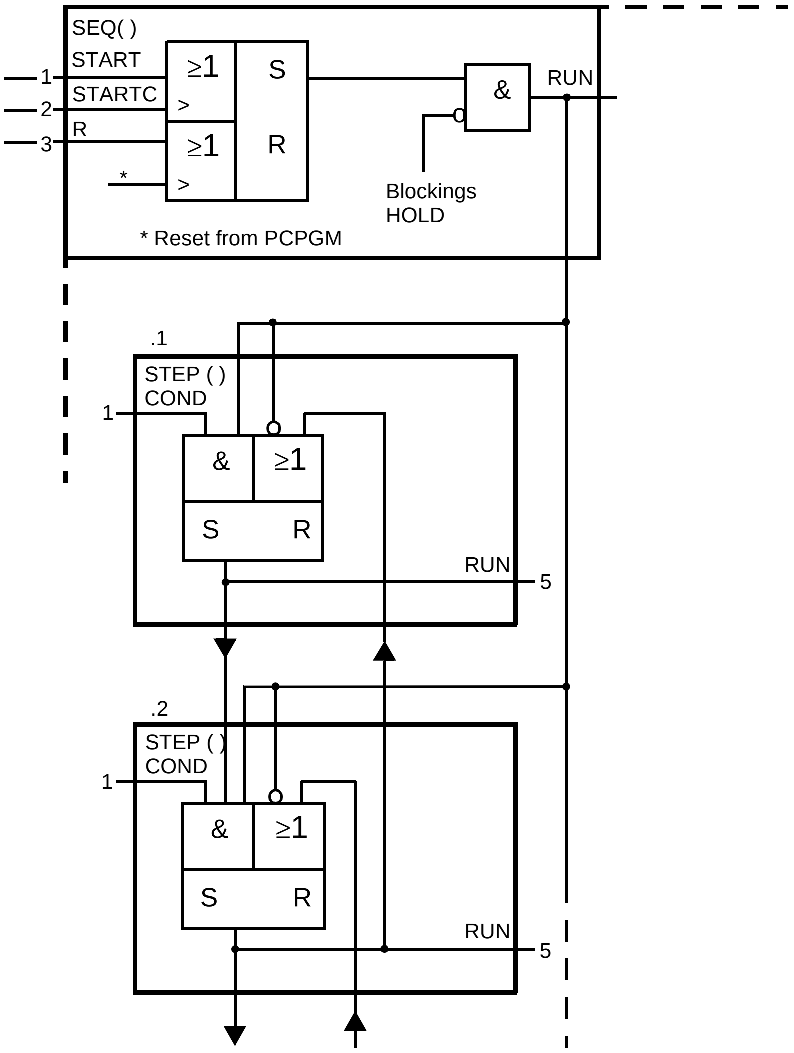


Figure 5-66. Function Diagram for Step Change

The sequence header has an internal memory function for the start signals. The memory is set using input START or STARTC and reset with the reset input R or with a superior reset signal from the PC program header.

Each step module has an internal memory function which is set when the step is activated, that is, when the condition input COND is 1 and the previous step is active. This memory function is reset when the step is left or when the sequence header is reset.

Step Change

The step is normally executed in a predetermined order. First, the step header is executed, then the PC elements of the step, and finally, the system checks the condition for step change.

The flow diagram in Figure 5-67 shows the step change between two steps,
step N and N + 1.

When the condition for step change from step N to N + 1 is fulfilled, the RUN output of step N is reset to 0 and the elements in the step are executed once in normal mode.

Following this, during the same cycle, the RUN output of step N + 1 is set to 1 and the PC elements of this step are executed for the first time. During this first execution of a step, the condition for step change to step N + 2 is ignored.
This implies that each step is activated during at least one cycle, that is, only one step change is performed per cycle.

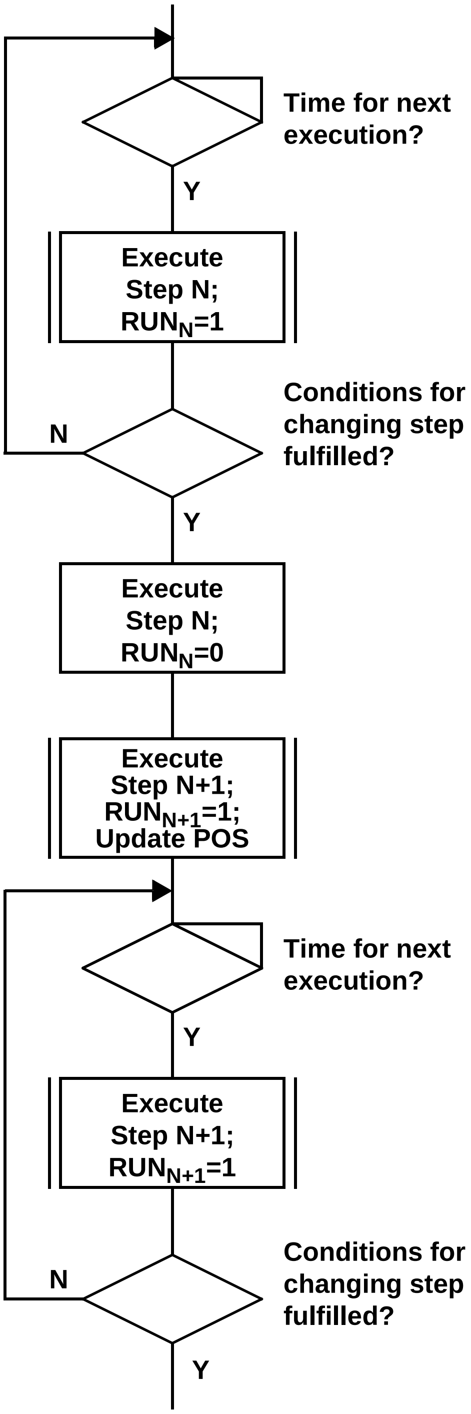

Time for next execution?
Y
Execute Step N; RUN$_N$=1
Conditions for changing step fulfilled?
N
Y
Execute Step N; RUN$_N$=0
Execute Step N+1; RUN$_{N+1}$=1; Update POS
Time for next execution?
Y
Execute Step N+1; RUN$_{N+1}$=1
Conditions for changing step fulfilled?
N
Y

Figure 5-67. Step Change

Figure 5-68 shows two steps where the counter included in each step is connected in two different ways. In the left-hand figure, the contents of the counter remain unchanged when the step is left. In the right-hand figure, the counter is cleared when the step is left since the RUN output of the step header is connected to the reset input of the counter.

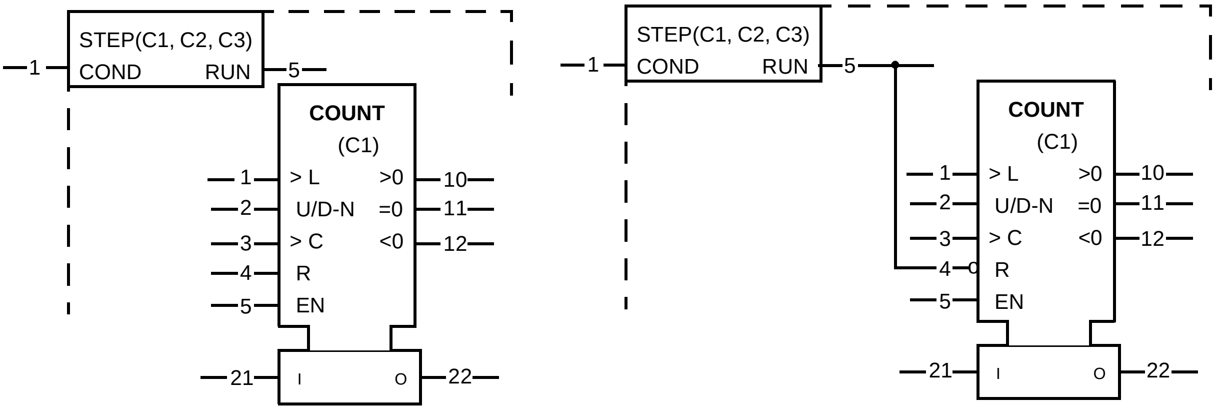


Figure 5-68. Example of Use of the RUN Signal

Use this feature when, for example, you want the data base variable MOTOR to be active even after the step is left. This can be implemented as in Figure 5-69.

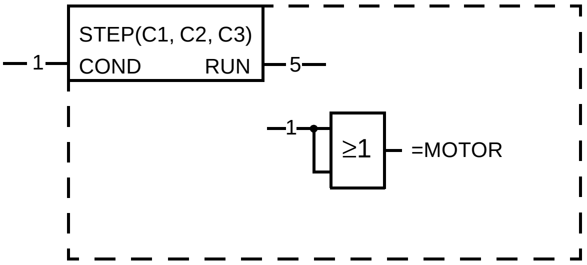


Figure 5-69. Output Whose Value Remains when the Step is Left

When you reset the entire sequence, however, all output data from all steps is cleared. In our example, the motor is consequently stopped when this occurs.

RUN Indications

The RUN output of the sequence header indicates that the sequence is activated and not blocked (HOLD also affects RUN).

The RUN output of the step header indicates that the step is active. This output can be used for direct control of an object by connecting it to the data base, see Figure 5-70. This output is not affected by HOLD or by blocking commands

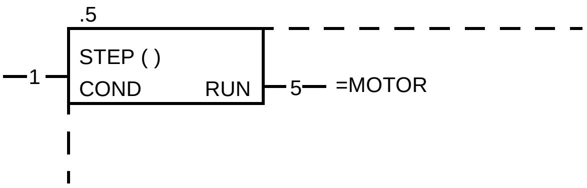


Figure 5-70. Connection of the RUN Output of a Step to the Data Base

In the example in Figure 5-70, the data base variable MOTOR will be 1 as long as step 5 is active, but will reset to 0 when the step is left.

Step Module Input and Output

Each step module has its own input (GET) and output (PUT) elements (refer to Section 11.3, Input and Output Phases). That means the following:

- Input from and output to the data base, as well as input and output for modules with differing priority (A, B, or C), is performed in the normal way.
- When changing steps, an extra execution with the RUN output = 0 is performed activating another data transfer in the same way.
- A step which has been left is no longer executed. This includes its input and output elements. All actual data values in the step are retained as they were following the last execution, with those values that were obtained with RUN = 0.

Chapter 6 Off-line Configuration and Programming

6.1 General

Off-line configuration and programming is the creation, modification, and target code generation of a user application program on an engineering station that is not connected to an controller system.

You need not connect the Advant Controller 100 Series station to the engineering station until the complete converted user application program must be downloaded to and started on controller.

This overview shows briefly how an user application program can be created, converted, and downloaded and then executed by the Advant Controller 100 Series controller.

For a detailed description of individual handling (windows, commands, type circuits, …), see the *Function Chart Builder User's Guide*.

With the support of the DB editor of the Function Chart Builder, you can edit a description of the I/O and communication modules and channels and of the common data. In case of AC 110, 160, and derivatives a special communication interface is the coupler to the Advant Fieldbus 100. This module, together with its related DSP and DAT elements, describes the connection to other AC 110, 160, 70, and derivative stations.

To generate a new user application program, edit the structure of the application program, beginning with the PCPGM, by means of the Function Chart Builder's PC editors in tree representation or in function chart representation.

In the function chart representation of the Function Chart Builder, you can edit the PC elements and their interconnections, as well as the interconnections to the DB elements.

After editing the complete application program, call the target code generator for converting the edited program to Advant Controller 100 Series target format and structure. Use the Function Chart Builder command **File | Generate Target Code**.

After converting the application program, connect the engineering station to the Advant Controller 100 Series system by the Function Chart Builder command **Target | Connect**.

Next start the download of the complete application program to the Advant Controller 100 Series system by the Function Chart Builder command **Target | Load Application**.

After downloading the converted application program, you can start the ADvant Controller 100 Series under user control by the Function Chart Builder command **Target | Deblock Program**.

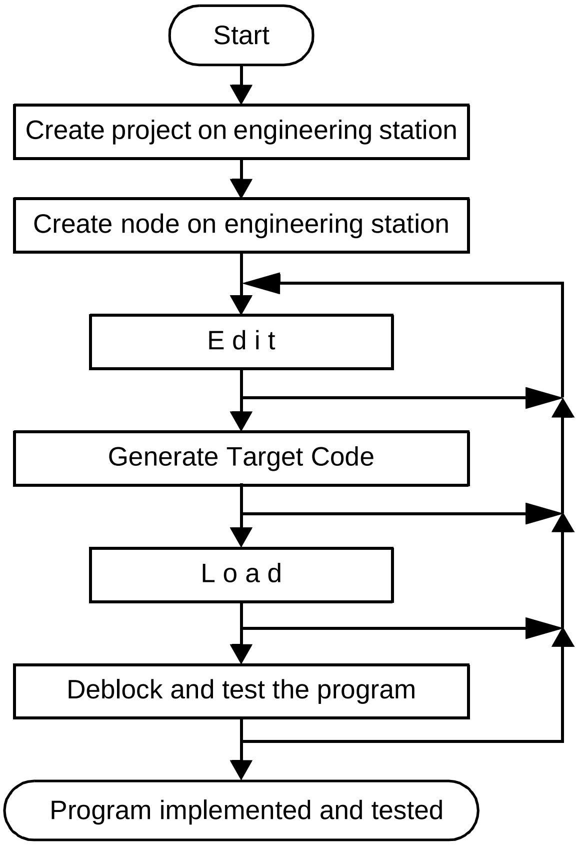

Start
Create project on engineering station
Create node on engineering station
Edit
Generate Target Code
Load
Deblock and test the program
Program implemented and tested

Figure 6-1. Implementation and Testing of a Program for AC 100 Series

6.2 Type Circuit Handling

Type circuit handling is supported for Advant Controller 100 Series systems during the off-line creation and modification of user application programs. (Since Function Chart Builder 4.7 also the on-line creation and deletion of simple type circuits is supported. These are either DB-only type circuits or PC-only type circuits containing no executable units.)

The Function Chart Builder offers different handling and on-line test capabilities depending on the structure and complexity of the type circuits:

- Instances of complex type circuits (such as the ones containing any of the structure PC elements CONTRM, MASTER, SEQ, SLAVEM, STEP), must be disintegrated.
  That means, these type circuits will not be referenced, but resolved in the data base or PC part of the application.
- Instances of simple type circuits (such as the ones containing only DB elements or the ones containing only PC elements, but no PC elements that are executable units), can be left in non-disintegrated form. That means these type circuits remain just referenced in the data base or PC part of the application.

Type circuit instances can be disintegrated by generating source code and then backtranslating source code with disintegration of type circuits.

6.3 Translate Functions

6.3.1 Conversion from/to AMPL Source Code

For a description of the generation and storage of a user application program, see Section 3.2, Generation and Storage.

The Function Chart Builder offers the commands **File | Generate Source** and **File | Backtranslate Source** for:

- Generation of AMPL source code from the FCB's internal data representation
- Backtranslation of AMPL source code to the FCB's internal data representation.

The scope of these commands can be:

- A complete node
- DB part
- All PC programs
- A single PC program
- Type circuits.

6.3.2 Generation of Target Code

The Function Chart Builder offers the command **File | Generate Target Code** for generation of target code corresponding to Advant Controller 100 Series format and structure from the FCB's internal data representation.

The scope of this command can be:

- A complete node (DB part and PC part)
- DB part, possibly containing disintegrated type circuits or references to type circuits depending on their structure and complexity
- PC part, possibly containing disintegrated type circuits or references to type circuits depending on their structure and complexity.

Generate target code for a complete node if you change the configuration part (for example, DB elements/channels, I/O modules, common data elements).

It is useful to generate target code for the DB part if you change the configuration part (for example, DB elements/channels, I/O modules, common data elements), for test purposes. If you wish to download your user application program, you must also convert the PC part.

Generate target code for the PC part if you change the programming part (for example, PC elements, connections).

In case of AC 100 Series nodes supporting application retrieval (upload and backtranslation of target code), you can generate also additional, complete information for storing in and later retrieval from the target.

You can fill also the Bus Configuration Database and generate BOIL*.C00 and COIL*.C00 files to support Human-Machine Interface systems and to facilitate access of process signals in AC 100 Series targets connected to Advant Fieldbus 100.

6.3.2.1 Restrictions

There are different restrictions when you are generating target code. Some of them are fixed and some depend on the actual combination of elements.

Fixed Restrictions

Fixed restrictions are shown in Table 6-1.

Table 6-1. Fixed Restrictions

Parameter	Limit
Maximum number of executable units	32
Maximum number of PC programs (PCPGM)	1
Maximum number of control modules (CONTRM + MASTER + SEQ)	31
Maximum number of control modules to be activated at initialization	1
Maximum number of control modules to be activated at warm-start	1
Maximum number of PC elements per executable unit	2000
Maximum number of PC element terminals per executable unit	7000
Maximum number of exchange variables between executable units	2000

Integer Long Values

On the engineering station, the maximum range for Integer Long (IL) is -2147483647 to 2147483646.

6.3.2.2 In-line Code

In order to achieve a very fast execution of binary operators, these operators are not handled like normal function block calls. They are coded within the object code as directly executable processor instructions.

Binary operators which have no other binary operator preceding them and no other binary operator following them are not in-line coded.

The PC elements AND, OR, XOR, and SR are realized as in-line code.

6.3.2.3 Target Code Generation Procedure

1. Generate target code by the **File | Generate Target Code** command.

   The Function Chart Builder opens the **Output** window and protocols the target code generation here and also in the \DBDATA\COTL0000.C00 and \PCDATA\RETL0000.C*pp* files (where *pp* = PC program number).

2. Evaluate the target code generation protocol.

   a. Check the lines "`w Warnings detected`" and "`e Errors detected`" in both sections (files).

      w>0 indicates violation of programming conventions, for example, the case when input connections are left unconnected.

      e>0 indicates violation of normal execution conditions on the target, for example, missing necessary DB elements or exhaustion of resources (RAM, PROM, and code and data area).

   b. In case of exhaustion of resources identify the violated requirement and the section(s) and the executable unit(s) responsible for the violation.

      The estimated values of memory requirements delivered by the **PC-Section | Memory Calculation | …** commands are also helpful.

3. If errors, warnings, or both occurred, change your application accordingly.
   That is, create the missing DB elements, restructure your application to decrease resource demand, connect open input PC terminals, and so on.

   Then repeat the procedure from Step 1.

6.3.2.4 Memory Requirements on the Target

1. Sum of PROM required by sections of the application must be less than the free PROM on the target.

   required_PROM = DB_section_PROM + PC_section_PROM

   (This also equals *sum_DB_section_CODE* + *sum_PC_section_CODE*.)

   Find free PROM in the Technical Description of the corresponding Advant Controller.

2. Sum of RAM required by sections of the application must be less than the free RAM on the target.

   required_RAM = DB_section_RAM + PC_section_RAM

   (This also equals *sum_DB_section_CODE* + *sum_PC_section_CODE* +
   + *sum_DB_section_DATA* + *sum_PC_section_DATA*.)

   To check free RAM choose **Target | Clear PROM**, then **Target | Restart**.

   Find free RAM in the **Free Memory** row of the (last) **Status Report**.

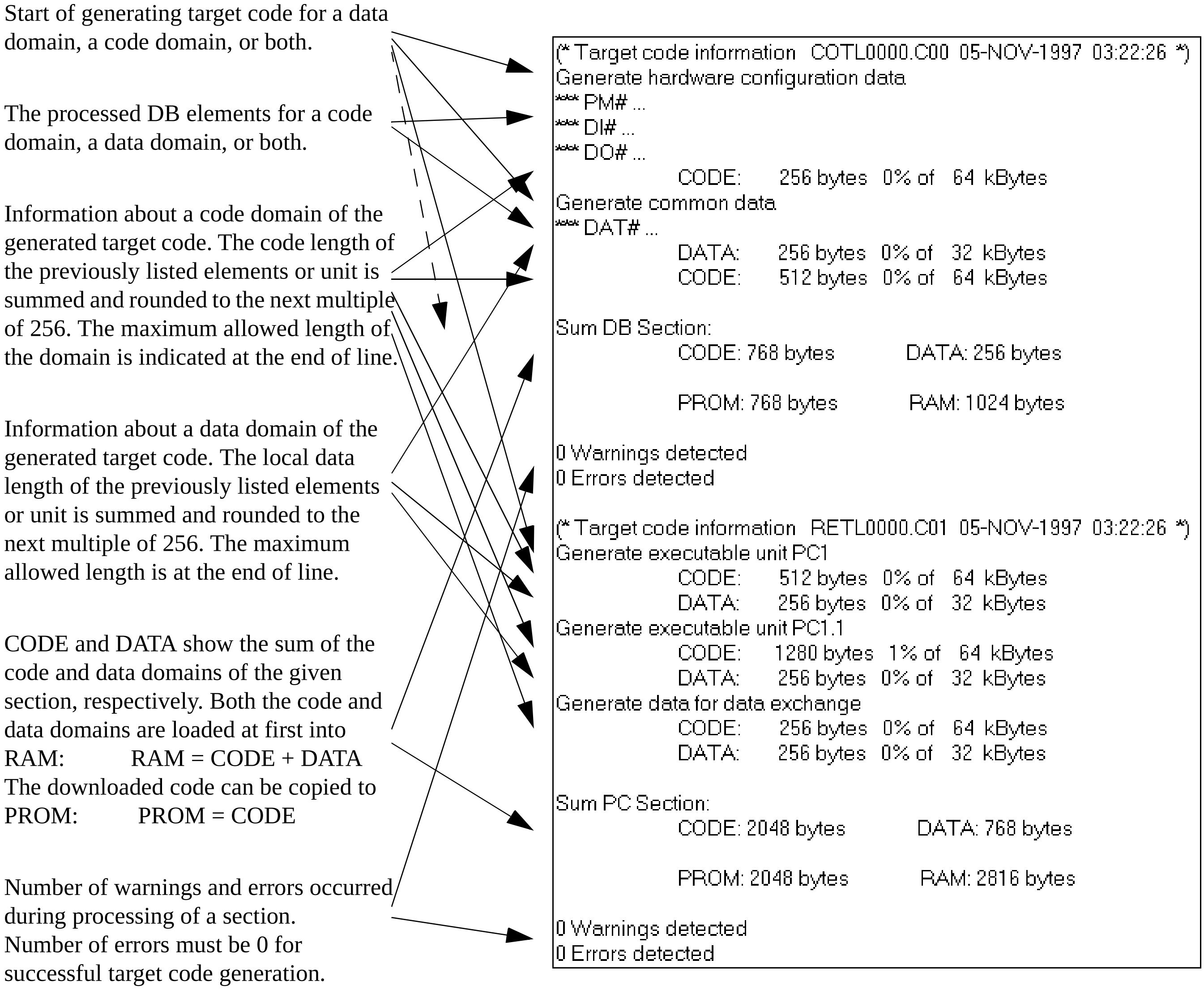
3. Sum of code of each code domain must be less than 64 kBytes.
   The code domains are denoted by `CODE:` in the target code generation protocol.
   In case of application retrieval support also Complete Info *xxx* domains may be generated.
   With respect to memory requirements, they count as code domains.

4. Sum of data area of each data domain must be less than the maximum indicated at the end of corresponding lines in the target code generation protocol.

   The maximum length of data domains is normally 32 kBytes. But in some cases it is 8 kBytes, for example, in case of PARDAT elements for AC 110 2.1.

   The data domains are denoted by `DATA:` in the target code generation protocol.

6.3.2.5 Example of Target Code Generation Protocol



Start of generating target code for a data domain, a code domain, or both.

The processed DB elements for a code domain, a data domain, or both.

Information about a code domain of the generated target code. The code length of the previously listed elements or unit is summed and rounded to the next multiple of 256. The maximum allowed length of the domain is indicated at the end of line.

Information about a data domain of the generated target code. The local data length of the previously listed elements or unit is summed and rounded to the next multiple of 256. The maximum allowed length is at the end of line.

CODE and DATA show the sum of the code and data domains of the given section, respectively. Both the code and data domains are loaded at first into
RAM: RAM = CODE + DATA
The downloaded code can be copied to
PROM: PROM = CODE

Number of warnings and errors occurred during processing of a section. Number of errors must be 0 for successful target code generation.

```
(* Target code information  COTL0000.C00  05-NOV-1997  03:22:26  *)
Generate hardware configuration data
*** PM# ...
*** DI# ...
*** DO# ...
             CODE:      256 bytes   0% of    64  kBytes
Generate common data
*** DAT# ...
             DATA:      256 bytes   0% of    32  kBytes
             CODE:      512 bytes   0% of    64  kBytes

Sum DB Section:
             CODE: 768 bytes            DATA: 256 bytes

             PROM: 768 bytes            RAM: 1024 bytes

0 Warnings detected
0 Errors detected

(* Target code information  RETL0000.C01  05-NOV-1997  03:22:26  *)
Generate executable unit PC1
             CODE:      512 bytes   0% of    64  kBytes
             DATA:      256 bytes   0% of    32  kBytes
Generate executable unit PC1.1
             CODE:     1280 bytes   1% of    64  kBytes
             DATA:      256 bytes   0% of    32  kBytes
Generate data for data exchange
             CODE:      256 bytes   0% of    64  kBytes
             DATA:      256 bytes   0% of    32  kBytes

Sum PC Section:
             CODE: 2048 bytes           DATA: 768 bytes

             PROM: 2048 bytes           RAM: 2816 bytes

0 Warnings detected
0 Errors detected
```

Figure 6-2. Target Code Generation Protocol (Example)

NOTES

In case of executable unit code domains a further 256 bytes are added for internal code to code length.
The length of data exchange internal code is 256 bytes.
The local data length of data exchange between executable units depends on the number and data type of these connections.

The values delivered by the **PC-Section | Memory Calculation | …** commands do not comprise these code and data lengths.

Chapter 7 Connecting a Controller to an Engineering Station

7.1 Connect

7.1.1 About Target Connection

Purpose

To connect the engineering station to an AC 100 Series target.

Definition

You connect the engineering station to an AC 100 Series target mainly in two ways:

- Directly
- By routing via other AC 100 Series or AC 400 Series nodes of a network.

NOTE

Different AC 100 Series node types support different cases of routing.

The engineering station to be connected can be the local one, on which you run the application requesting the connection, such as the Function Chart Builder or the Diagnostics program, but also another remote one, a server, connected to a common computer network. Then your local engineering station (“this PC”) is client of the connection service available on the server.

Conditions of Successful Connection

1. The connection service is available either on the local engineering station or on a server connected to a common computer network, which you intend to use.
   For conditions of successful server operation and instructions for the Windows NT system administrators see *Function Chart Builder User’s Guide*, Appendix G, Configuration of AdvaBuild Communicator.
2. If an engineering board is involved in the routing, the On-line Builder must be installed on the pertaining engineering station and the engineering board must be configured appropriately by the Engineering Board Setup program.
3. The engineering station(s), the involved nodes along the routing path, and the target are cabled appropriately.
4. Applications, loaded to the nodes along the routing path, are configured to support the communication via MB 300, AF 100, and RCOM net, as applicable.
5. The connection path is defined appropriately in the **Connect** dialog of the **Target | Connect** command.

7.1.2 Connect the Engineering Station to an AC 100 Series Target

Preparation of Engineering Station, Links, and Intermediate Nodes

See Section 2.4.5, Preparation of Connection.

Connection

1. Choose the **Target | Connect** command.
   The Function Chart Builder displays the **Connect** dialog box.
2. In case of simple and repeated connections select the appropriate connection path in the connection history contained in the **Connection Path** drop-down list box without bothering about details.
   Or, in case of complex paths and new connections, activate the **Connect Assistant** by the **Assistant** button.

   The **Connect Assistant** guides you through steps of connection path definition:

   a. Define the server, that is, the engineering station offering the connection service.
      This can be “this PC” but also some other PC accessible via a computer network.
      To select a remote server, you have three possible ways:
      - Activate the **Browse** button and find a server in your computer network.
      - Select one of the recently used servers in the **Selected server PC** drop-down list box.
      - Enter a server directly in **Selected server PC** text box.
        You can do the latter in different forms, like
        - Name corresponding to universal naming convention (UNC),
          for example, `\\gpa789 or gpa789`
        - TCP-IP address, for example, `123.456.789.87`
        - Distributed server name (DSN) like a world-wide-web address,
          for example, `www.abb.de` or `deind.de`.

      Select also the server's interface or port to be used.

   b. Define the target by its AF 100 or RCOM address if it has one.
      The address has to correspond to the setting by appropriate DB elements of the application running on the target.
      An AF 100 address consists of bus number, station number, and position.
      An RCOM address consists of remote net number, remote node number, and position.
      In case of point-to-point COMx connection, there is no need to define the address explicitly.

   c. Define the intermediate nodes, if there are any, by their MB 300 or AF 100 address.
      The address has to correspond to the setting by appropriate DB elements of the application running on the intermediate node.
      An MB 300 address consists of net number and node number.
      An AF 100 address consists of bus number, station number, and position.
      In case of point-to-point X3 connection, there is no need to define the address explicitly.

   d. Return to the **Connect** dialog.

3. Activate the **OK** button.
   The connection will be established. It may take some seconds.
   The the Function Chart Builder requesting the connection can continue. It displays the **Status Report** and the **Options Report** about the connected target.

7.1.3 Types of Routing, Diagrams

The following types of routing are supported:

1. Point-to-point connection via COMx (no routing)
2. Routing via COMx
   a. Routing via COMx, AC 100 Series Node, AF 100
   b. Routing via COMx, AC 100 Series Node, AF 100, 2nd AC 100 Series Node, 2nd AF 100.
3. Routing via X3
   a. Routing via X3, AC 4xx, AF 100
   b. Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100
   c. Routing via X3, AC 4xx, RCOM Net
   d. Routing via X3, Several AC 4xx Connected by MB 300, AF 100
   e. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100
   f. Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net.
4. Direct Connection via AF 100 (no routing)

See the topology of routing with exemplary addresses for each type of routing in the following.

The pertaining settings of connection path in the **Connect** dialog of the **Target | Connect** command refer

- **Target Name**
  Is set on the basis of the status information fetched from the target after a successful connection.
- **Server**
  The empty string ('') and “This PC” denotes your local engineering station you work on. If you want to use the connection service running on a remote engineering station, you can set it with the help of the Connect Assistant.
- **PC Interface**
  This is the interface of the server towards the routing network (in case of routing) or towards the target (in case of direct connection).
  Examples: COM1, COM4, X3, AF100IF.
- **Intermediate Node Address**
  This is the identification of the intermediate node.
  In case of point-to-point (direct) connection it is omitted. In other cases the intermediate node is identified by the kind of bus it is reached and by its address on this bus.

Examples of address form: `AF100[`*`bus, station, position`*`]`, `MB300[`*`net, node`*`]`, `RCOM[`*`rnet, rnode, position`*`]`.

In the **Connect Assistant**, **Position** of the intermediate node is preset originally to 0. Set the appropriate value according to the configuration of the corresponding AC 100 Series station:

Table 7-1. Position of Intermediate Nodes

Node Type	Position
AC 110	1
AC 70	0
AC 160 (station with CPU redundancy)	3, 4,… 8

- **Target Address**
  This is the identification of the target.
  In case of point-to-point (direct) connection it is omitted.
  In other cases the target is identified by the kind of bus it is reached and by its address on this bus.
  Examples of address form: `AF100[`*`bus, station, position`*`]`, `RCOM[`*`rnet, rnode, position`*`]`.

  In the **Connect Assistant**, **Position** of the target is preset on the basis of the corresponding value for the target in the Application Builder.

Table 7-2. Position of Target

Node Type	Position
AC 110	1
AC 55	0
AC 70	0
AC 160 (station with CPU redundancy)	3, 4,… 8

Point-to-point Connection via COMx

Topology of connection with exemplary addresses:

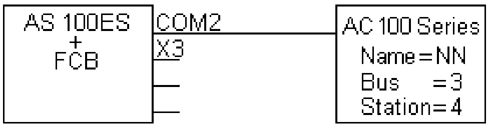
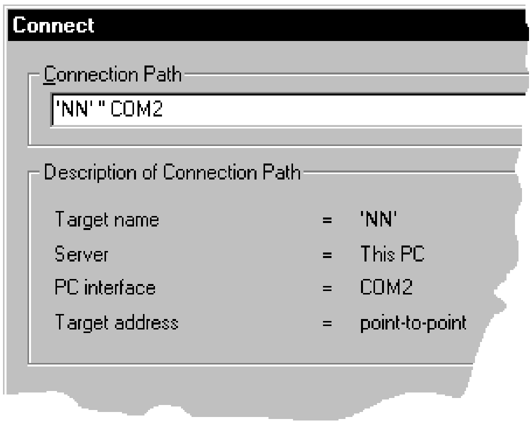


Figure 7-1. Point-to-point Connection via COMx, Topology

Detail of **Connect** dialog with corresponding setting:



Connect

Connection Path

'NN' " COM2

Description of Connection Path

Target name = 'NN'
Server = This PC
PC interface = COM2
Target address = point-to-point

Figure 7-2. Point-to-point Connection via COMx, Setting

Routing via COMx, AC 100 Series Node, AF 100

Topology of connection with exemplary addresses:

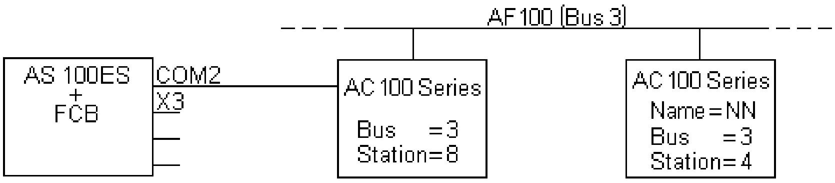
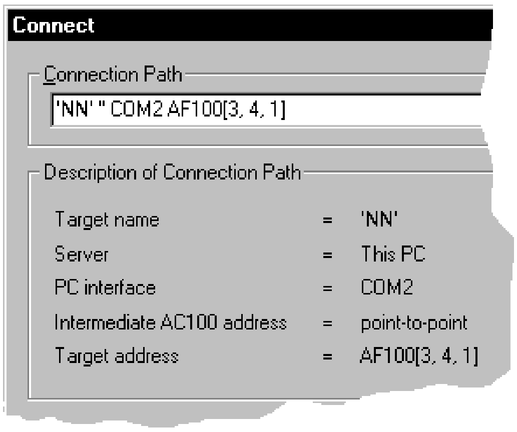


Figure 7-3. Routing via COMx, AC 100 Series Node, AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connect

Connection Path

'NN' " COM2 AF100[3, 4, 1]

Description of Connection Path

Target name = 'NN'
Server = This PC
PC interface = COM2
Intermediate AC100 address = point-to-point
Target address = AF100[3, 4, 1]

Figure 7-4. Routing via COMx, AC 100 Series Node, AF 100, Setting

Routing via COMx, AC 100 Series Node, AF 100, 2nd AC 100 Series, 2nd AF 100

Topology of connection with exemplary addresses:

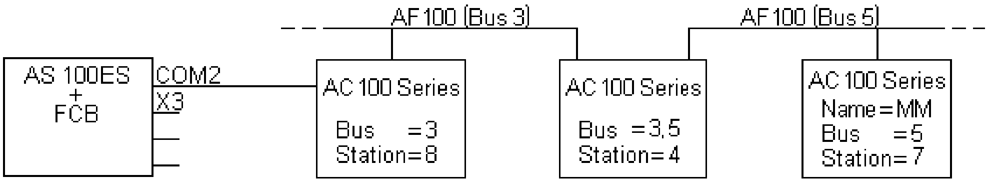
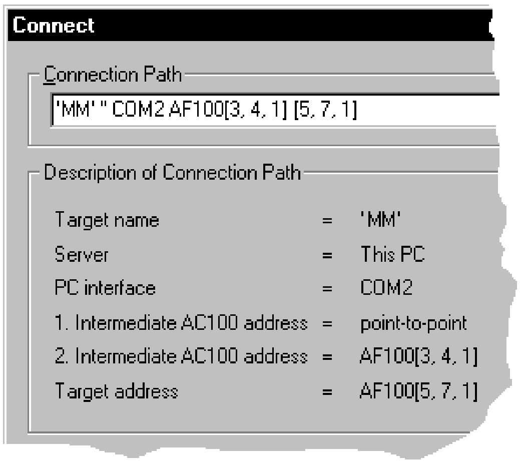


Figure 7-5. Routing via COMx, AC 100 Series, AF 100, 2nd AC 100 Series, 2nd AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connection Path
'MM' " COM2 AF100[3, 4, 1] [5, 7, 1]

Description of Connection Path

Target name = 'MM'
Server = This PC
PC interface = COM2
1. Intermediate AC100 address = point-to-point
2. Intermediate AC100 address = AF100[3, 4, 1]
Target address = AF100[5, 7, 1]

Figure 7-6. Routing via COMx, AC 100 Series, AF 100, 2nd AC 100 Series, 2nd AF 100, Setting

Routing via X3, AC 4xx, AF 100

Topology of connection with exemplary addresses:

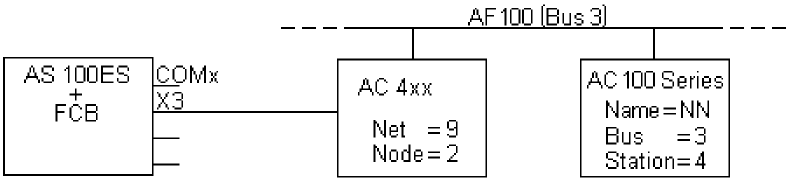
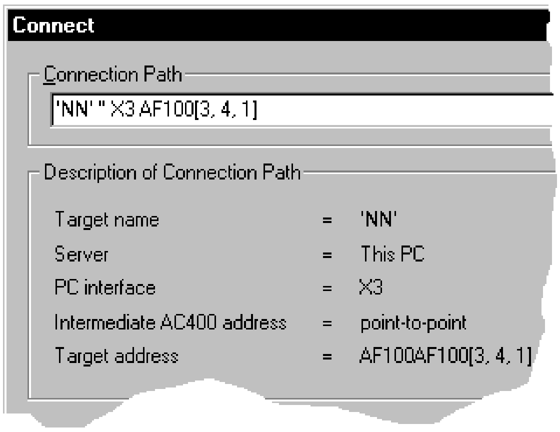


Figure 7-7. Routing via X3, AC 4xx, AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connection Path

'NN' " X3 AF100[3, 4, 1]

Description of Connection Path

Target name = 'NN'
Server = This PC
PC interface = X3
Intermediate AC400 address = point-to-point
Target address = AF100AF100[3, 4, 1]

Figure 7-8. Routing via X3, AC 4xx, AF 100, Setting

Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100

Topology of connection with exemplary addresses:

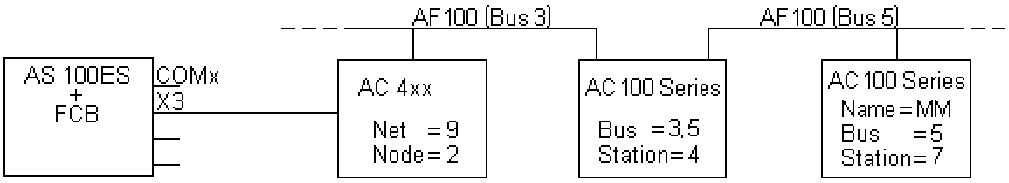
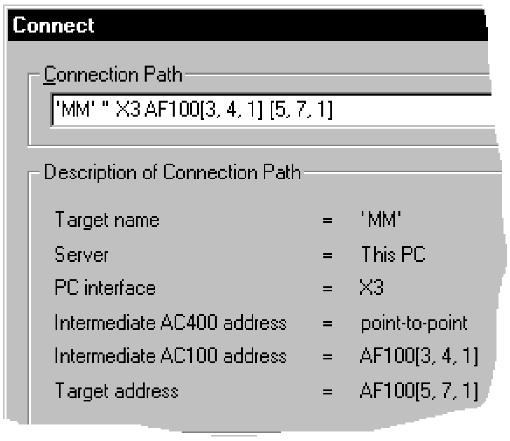


Figure 7-9. Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connection Path

'MM' " X3 AF100[3, 4, 1] [5, 7, 1]

Description of Connection Path

Target name = 'MM'
Server = This PC
PC interface = X3
Intermediate AC400 address = point-to-point
Intermediate AC100 address = AF100[3, 4, 1]
Target address = AF100[5, 7, 1]

Figure 7-10. Routing via X3, AC 4xx, AF 100, AC 100 Series Node, 2nd AF 100, Setting

Routing via X3, AC 4xx, RCOM Net

Topology of connection with exemplary addresses:

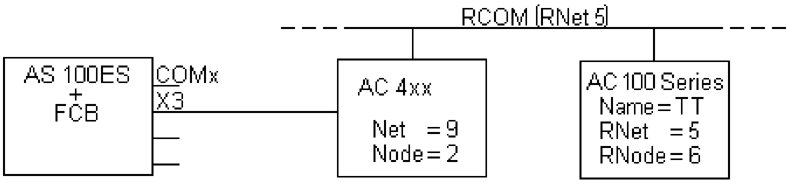
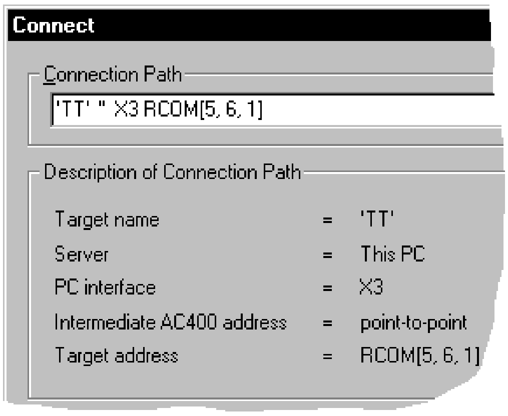


Figure 7-11. Routing via X3, AC 4xx, RCOM Net, Topology

Detail of **Connect** dialog with corresponding setting:



'TT' " X3 RCOM[5, 6, 1]

Target name = 'TT'
Server = This PC
PC interface = X3
Intermediate AC400 address = point-to-point
Target address = RCOM[5, 6, 1]

Figure 7-12. Routing via X3, AC 4xx, RCOM Net, Setting

Routing via X3, Several AC 4xx Connected by MB 300, AF 100

Topology of connection with exemplary addresses:

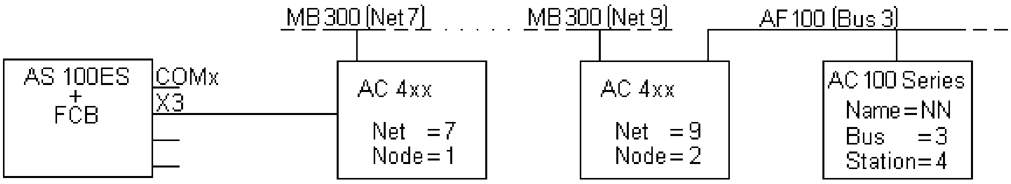
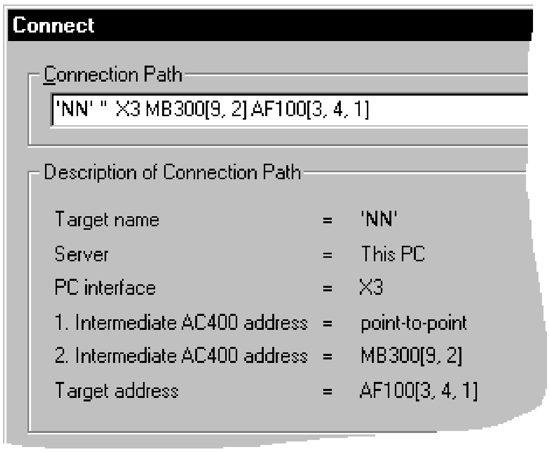


Figure 7-13. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connection Path

'NN' '' X3 MB300[9, 2] AF100[3, 4, 1]

Description of Connection Path

Target name = 'NN'
Server = This PC
PC interface = X3
1. Intermediate AC400 address = point-to-point
2. Intermediate AC400 address = MB300[9, 2]
Target address = AF100[3, 4, 1]

Figure 7-14. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, Setting

Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100

Topology of connection with exemplary addresses:

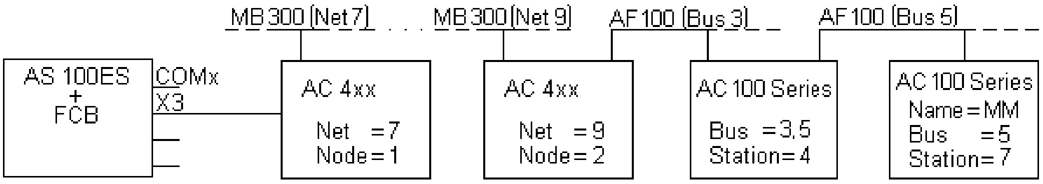
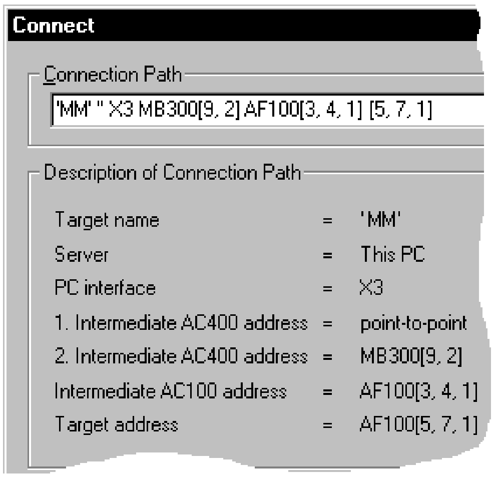


Figure 7-15. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



Connection Path

'MM' '' X3 MB300[9, 2] AF100[3, 4, 1] [5, 7, 1]

Description of Connection Path

Target name = 'MM'
Server = This PC
PC interface = X3
1. Intermediate AC400 address = point-to-point
2. Intermediate AC400 address = MB300[9, 2]
Intermediate AC100 address = AF100[3, 4, 1]
Target address = AF100[5, 7, 1]

Figure 7-16. Routing via X3, Several AC 4xx Connected by MB 300, AF 100, AC 100 Series Node, 2nd AF 100, Setting

Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net

Topology of connection with exemplary addresses:

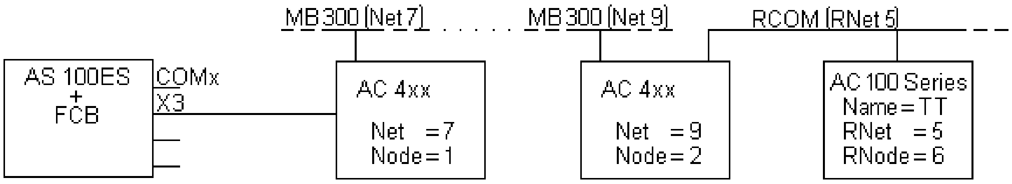
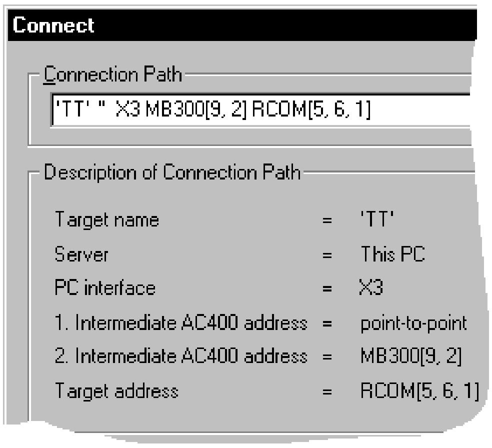


Figure 7-17. Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net, Topology

Detail of **Connect** dialog corresponding setting:



'TT' '' X3 MB300[9, 2] RCOM[5, 6, 1]

Target name = 'TT'
Server = This PC
PC interface = X3
1. Intermediate AC400 address = point-to-point
2. Intermediate AC400 address = MB300[9, 2]
Target address = RCOM[5, 6, 1]

Figure 7-18. Routing via X3, Several AC 4xx Connected by MB 300, RCOM Net, Setting

Direct Connection via AF 100

Topology of connection with exemplary addresses:

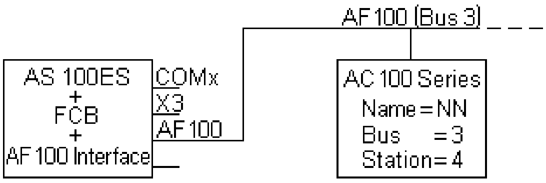
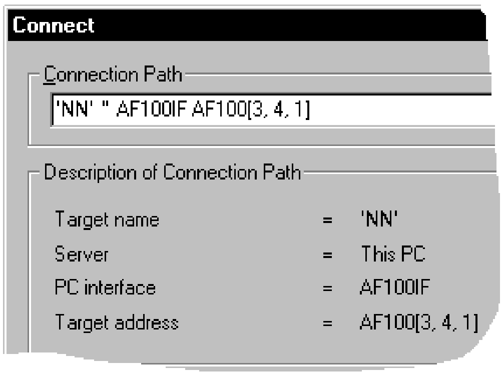


Figure 7-19. Routing via AF 100, Topology

Detail of **Connect** dialog with corresponding setting:



'NN' " AF100IF AF100[3, 4, 1]

Target name = 'NN'
Server = This PC
PC interface = AF100IF
Target address = AF100[3, 4, 1]

Figure 7-20. Routing via AF 100, Setting

7.2 Target Access Protection

NOTE

These functions are available only for targets, the base software of which supports target access protection. Such node types are, for example, AC 110 2.3, AC 160 1.2, and AC 166 1.0.

The Function Chart Builder offers three different target access protection functions:

1. Target password protection from manipulations of the application program by unauthorized persons

   In fact, the connection to a target with a certain application program running on it is protected by the target password.

2. Application program checksum

   You can inspect and store the checksum (CRC) of the application program running in the RAM and inspect the checksum of the system in target PROM.
   These checksums can be inspected later again and be compared with stored values.

   If they are not equal, the application has been changed in the meantime.

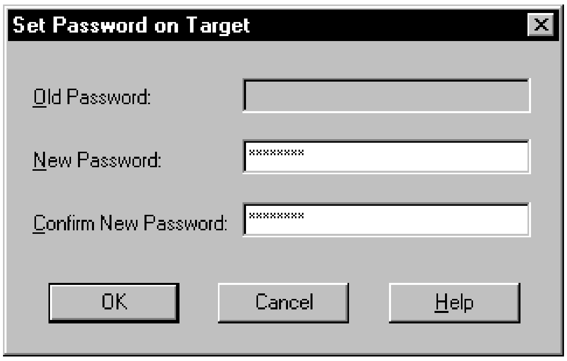
3. Restriction of network programming

   You can allow or forbid programming of a target by routing via an AF 100 network.

Target Password Protection

1. At cold start of the target or stop target the initial target password is empty and the connection to the target is not protected.

2. You can define a target password by the **Target | Set Password** command if the target is connected.



New Password: *********
Confirm New Password: *********

Figure 7-21. Set Initial Password on Target

Enter here the password in the **New Password** and **Confirm New Password** text boxes.

The password has to consist of 8 to 50 printable characters, it is case sensitive.

On activating the **OK** button the new password is saved in the target RAM.

CAUTION

Use password protection only if it is reasonable.
Be aware, if you forget the password, there is no “universal password”, you can just stop the target to clear the password.
A warm start does not clear the password.

3. After that, each time you—or any other user—connect to this target, the Function Chart Builder prompts to enter the password associated with the target.

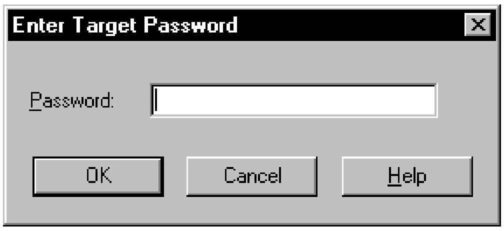


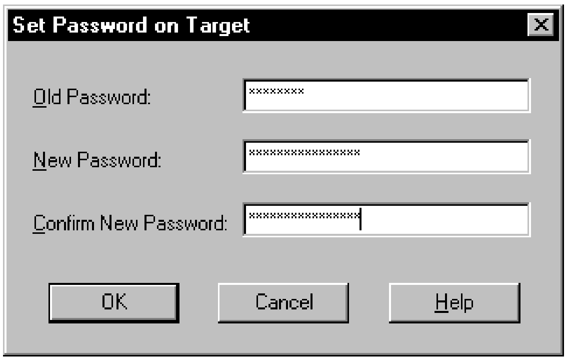
Figure 7-22. Target Password Query at Target Connection

If you enter the correct password, the Function Chart Builder connects to the target and offers its on-line testing and editing. You can test and edit the application stored in the target.

If you enter an incorrect password, the Function Chart Builder refuses the connection. You can neither test nor edit the application program stored in target, just try to enter the correct password again.

4. You can change or clear the target password by the **Target | Set Password** command if the target is connected.

   In these cases you have to enter at first the old password:



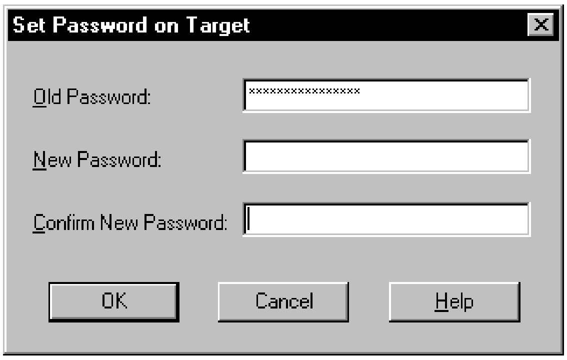
Old Password: xxxxxxxx

New Password: xxxxxxxxxxxxxxxx

Confirm New Password: xxxxxxxxxxxxxxxx

Figure 7-23. Change Password on Target

You clear the password by the empty string as new password:



Old Password: xxxxxxxxxxxxxxxx

New Password:

Confirm New Password:

Figure 7-24. Clear Password on Target

5. The target password is implicitly cleared at stopping the target by reset button and no start position of the stop mode switch.

   Together with the application program in the target RAM, the target password, if there is any, is implicitly copied to the target PROM

   The target password is implicitly taken from the target PROM at cold start of the target by reset button and cold start position of the stop mode switch.

NOTE

Do not mix up the target password with the ODB password.
See Section 3.3, Application Protection.

Nevertheless, if the ODB password equals the target password, you do not need to enter the same password again when connecting to the target.

Application Program Checksum

Use the **Target | Application Checksum** command to

- Inspect the actual CRC of the application in the target RAM, the stored CRC, and date and time of CRC storage, and the CRC of the system in target PROM
- Store the application checksum (CRC) in the target RAM.

NOTE

You can access these three CRC values also by an appropriate PC element and use them to check integrity or change of your application program and PROM contents.

Remote Access Protection - Restriction of Network Programming

To restrict remote accessing and programming of targets by routing via MB 300, AF100, and RCOM networks, the DB elements of processor modules PM*nnn* of appropriate nodes offer the terminal NETACC.

If you program NETACC=1 (default), the target can be connected and programmed also remotely, by routing via MB 300, AF100, and RCOM networks.
If you program NETACC=0, the target can only be connected point-to-point via COM*x* and programmed directly. When the inadvertent user tries to connect the target remotely, the Function Chart Builder refuses the connection and shows the following warning message:

```
“This target is protected from being connected via the control
 network.
 You can only connect directly to the target using port COMx.”
```

7.3 Disconnect

Choose the command **Target | Disconnect** to terminate the session with the AC 100 Series station.

Chapter 8 Target Control Functions

8.1 General

On the engineering station, the Function Chart Builder offers commands in the **Target** menu for the following functions:

- Execution control of PC programs
- Maintenance features of the controller
- Maintenance features of the application.

Loading the application with the command **Target | Load Application** is the first step toward establishing an application on the Advant Controller 100 Series target.

Start the execution of the application with the command **Target | Deblock Program**.
Stop the execution with the command **Target | Block Program**.

Trigger a cold start with the command **Target | Restart**.

Back up the application on a flash PROM with the command **Target | Save in PROM**. Delete the contents of the flash PROM with **Target | Clear PROM**.

The **Target | Set Date and Time** command and the features of reporting status, errors, and options are helpful in maintaining an AC 100 Series target.

8.2 Load Application

The command **Target | Load Application** downloads a complete user application program to the connected AC 100 Series target.

The Function Chart Builder opens the **Output** window, protocols there the download process, and updates the **Status Report** of the target..

After the download is finished, the status of the target is updated.

If the download is successful, the Function Chart Builder offers switching to on-line mode.
See Section 9.1, About On-line and Off-line Modes and Mode Transitions.

An off-line generated user application program is downloaded to the target system as one unit. There is no need to download the DB and PC elements separately.

If the connected AC 110, 160, or 70 station is connected to an Advant Fieldbus 100 and the demanded target is not reachable, it is not possible to download directly by the connection cable to the AC 110, 160, or 70 station.

If the connected AC 110, 160, or 70 station is not connected to an Advant Fieldbus 100 and the station number specified within the already existing application program on the connected AC 110, 160, or 70 station differs from that of the connected engineering station, a warning is generated before the new application program is downloaded.

8.3 Block Program

The command **Target | Block Program** stops the user application program of the connected AC 100 Series target.

Before starting the function, confirm the desired action.

8.4 Deblock Program

The command **Target | Deblock Program** starts the user application program of the connected AC 100 Series target.

Before starting the function, confirm the desired action.

8.5 Restart

The command **Target | Restart** triggers a cold start of the connected AC 100 Series target.

Before starting the function, confirm the desired action.

8.6 Save In PROM

By the command **Target | Save in PROM**, the AC 100 Series target backs up the current user application program from RAM to flash EPROM.

After every cold-start of the AC 100 Series system, the application program from flash PROM is copied to RAM and started there.

Before starting the function, confirm the desired action.

8.7 Clear PROM

The command **Target | Clear PROM** deletes the contents of the flash PROM. For safety reasons, the command does not work if the application is in deblocked mode.

Before starting the function, confirm the desired action.

8.8 Set Date and Time

The command **Target | Set Date and Time** displays the actual date and time of the engineering station host system and the actual date and time of the target system. Set the date and time on the AC 100 Series according to the date and time on the engineering station. Since Function Chart Builder 4.7, you can also preset any time and date independent of the engineering station.

Before starting the set function, confirm the desired action.

8.9 Update Slave

The application program in the AC 100 Series targets is usually stored in RAM.
Use the **Target | Update Slave** command to copy the application program from the on-line connected master controller of an AC 100 Series target with CPU redundancy to the slave controller and save it there in non-volatile flash PROM.

This command is available only for target types supporting this function, such as AC 160.

Before starting the set function, confirm the desired action.

8.10 Upload Application

Use the **Target | Upload Application** command to upload the target code of an application program from the RAM of the on-line connected AC 100 Series target.

Application retrieval is supported by Function Chart Builder 4.6 and higher and this command is available only for new node types explicitly supporting application retrieval, such as AC 80 2.0.

There are several conditions for the success and scope of retrieval:

- The application can only be retrieved if you work with the same Function Chart Builder version as the one with which its target code was generated.
- For successful retrieval, node type and element library versions used by the application to be retrieved must be available and installed in the corresponding release subdirectories. Or they must be stored with the target code on the target.
- The command will not be executed if the application is exactly the same in the target and in the Function Chart Builder, for example, after a **Target | Load Application**. After any substantial change **Target | Upload Application** becomes effective again.
- The scope of application retrieval depends on the information included in the target code stored on the target, that is, on the options that were selected at target code generation.

The Function Chart Builder stores the target code files *.C00 and *.C?? in the \CONTROL subdirectory of the node.

This command is normally followed by a **File | Backtranslate Target Code** command.

8.11 Set Password, Application Checksum

The commands **Target | Set Password** and **Target | Application Checksum** can be used for protection of the application on the target.

These functions are available only for targets, the base software of which supports target access protection. Such node types are, for example, AC 110 2.3, AC 160 1.2, and AC 166 1.0.

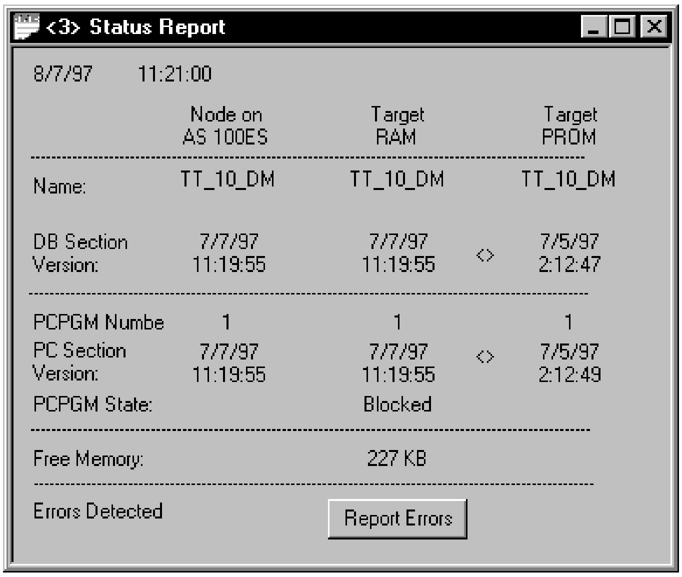
See Section 7.2, Target Access Protection.

8.12 Report Status

After connection to an AC 100 Series station, the engineering station automatically reports the status of the connected station in the **Status Report** window.

Choose the **Target | Status Report** command to get a new status report from the target.

See figure below for an example of the **Status Report** window:



```
8/7/97      11:21:00

                 Node on        Target          Target
                 AS 100ES       RAM             PROM
------------------------------------------------------------
Name:            TT_10_DM       TT_10_DM        TT_10_DM

DB Section       7/7/97         7/7/97          7/5/97
Version:         11:19:55       11:19:55   <>   2:12:47
------------------------------------------------------------
PCPGM Numbe      1              1               1
PC Section       7/7/97         7/7/97          7/5/97
Version:         11:19:55       11:19:55   <>   2:12:49
PCPGM State:                    Blocked
------------------------------------------------------------
Free Memory:                    227 KB
------------------------------------------------------------
Errors Detected
```

Figure 8-1. Status Report Window, Example

Compare the data.

For successful on-line operations node and application program descriptions must be identical in the **Node on AS 100ES** and in the **Target RAM** columns.

In case of errors you can start the Diagnostics (Diagnosis) program attached to the Function Chart Builder.

8.13 Report Options

The command **Target | Report Options** shows the selected options and the available options of the connected device. Compare the data.

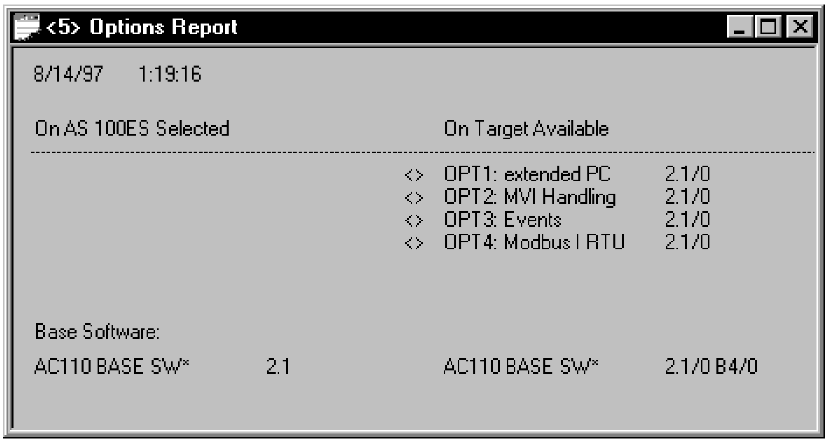
The options and the base software are identified in the following form:

ident_string *major_release* **.** *minor_release* / *revision*

The conditions of successful on-line operations are the following:

- The base software and all the options selected in the engineering station must be available in the on-line connected target.
- The *ident_string*, *major_release*, and *minor_release* of the item selected in the engineering station must equal the *ident_string*, *major_release*, and *minor_release* of the corresponding item on the target. So *revision* is not evaluated.

See figure below for an example of the **Options Report** window:

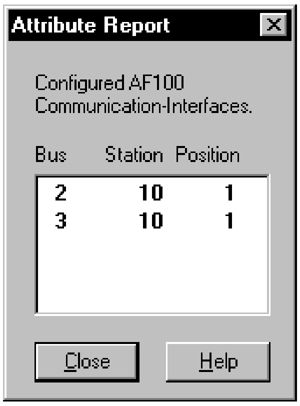


```
8/14/97     1:19:16

On AS 100ES Selected                          On Target Available
---------------------------------------------------------------------------
                                         <>   OPT1: extended PC      2.1/0
                                         <>   OPT2: MVI Handling     2.1/0
                                         <>   OPT3: Events           2.1/0
                                         <>   OPT4: Modbus I RTU     2.1/0


Base Software:

AC110 BASE SW*          2.1                   AC110 BASE SW*         2.1/0 B4/0
```

Figure 8-2. Report Options Window (Example, AC 55)

8.14 Report Attributes

Use the **Target | Report Attributes** command to read the attributes, such as the configured AF 100 communication interfaces, of the on-line connected target of a newer AC 100 Series node type, such as AC 160, AC 110 2.1 and higher, AC 80, AC 70 1.1 and higher, and AC 55 1.1, and show them in the following dialog box:



Configured AF100 Communication-Interfaces.

Bus	Station	Position
2	10	1
3	10	1

Figure 8-3. Report Attributes Dialog Box (Example, AC 110)

8.15 Report Errors

The command **Target | Diagnostics** starts the Diagnostics (Diagnosis) program attached to the Function Chart Builder and displays all errors reported from the connected AC 100 Series target in the **Error Report** window.

NOTE

The Diagnostics (Diagnosis) program is adapted to the different diagnostics features of both former and recent controller versions.

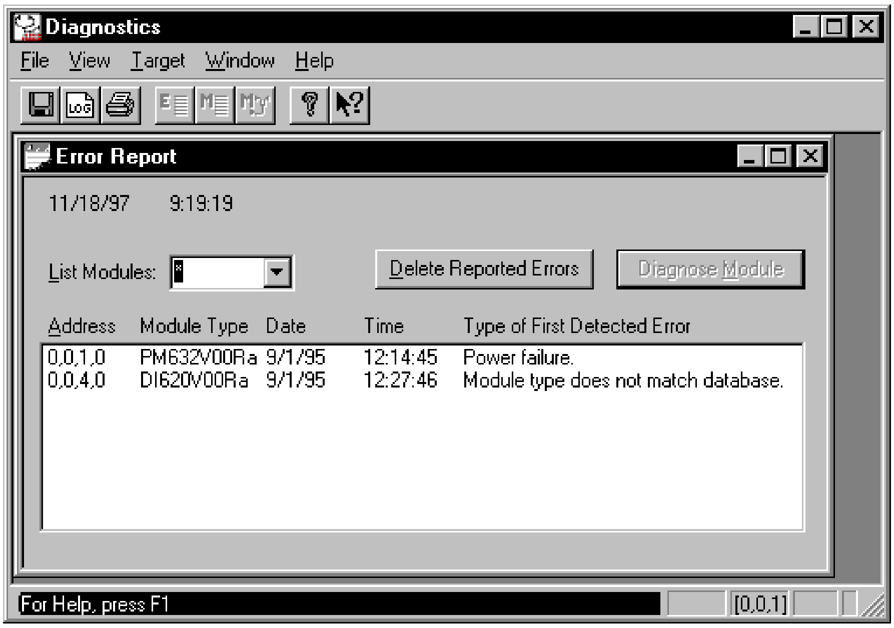
If you are working with an AC 100 Series target of a newer node type, such as AC 160, AC 110 2.1 and higher, AC 80, AC 70 1.1 and higher, and AC 55 1.1, this command starts the Diagnostics program attached to the Function Chart Builder.

If you are working with a target of node type AC 110 2.0, AC 70 1.0, and AC 55 1.0, this command starts the Diagnosis program attached to the Function Chart Builder and displays the reported errors in the Error Report window.

The Diagnostics (Diagnosis) program has its own menus and commands. For example, you can delete all displayed error messages on the AC 100 Series by the Diagnosis command **Target | Delete All Reported Errors**.

For the error texts and an explanation of the errors as well as corrective actions see *Advant Controller 110 User's Guide*, *Advant Controller 160 User's Guide*, *Advant Controller 70 User's Guide*, and *Advant Controller 55 User's Guide*.

See figure below for an example of the Diagnosis application window and the **Error Report** window:



11/18/97 9:19:19

Address	Module Type	Date	Time	Type of First Detected Error
0,0,1,0	PM632V00Ra	9/1/95	12:14:45	Power failure.
0,0,4,0	DI620V00Ra	9/1/95	12:27:46	Module type does not match database.

For Help, press F1 [0,0,1]

Figure 8-4. Report Error Window (Example, AC 110 2.0)

Chapter 9 On-line Configuration and Programming

9.1 About On-line and Off-line Modes and Mode Transitions

Definition

Generally, you prepare an application for an AC 100 Series target system using the Function Chart Builder off-line. Then, you connect the engineering station to the target and download the target code of the application. The latter can be time consuming.

Now, you can test and edit the application running on the target by the Function Chart Builder on-line. When editing on-line, you can change only one executable unit at a time. So, you have to download only the changed part of the target code.

You may want to prepare a change limited to one executable unit off-line and disconnected from the target, then connect and go on-line again and download only the difference to achieve a bumpless modification of the running application on the target.

To support these procedures, four modes (states) are available. When the Function Chart Builder works with the *same* AC 100 Series target system, it is any time in one of them:

- Off-line (non-connected)
- Off-line connected
- On-line (connected)
- On-line preparation.

NOTES

The on-line preparation mode is supported since Function Chart Builder release 4.6.

Modes and Transitions - State Diagram

Follow the different possible modes (states) of the Function Chart Builder working on a given AC 100 Series node in the following figure:

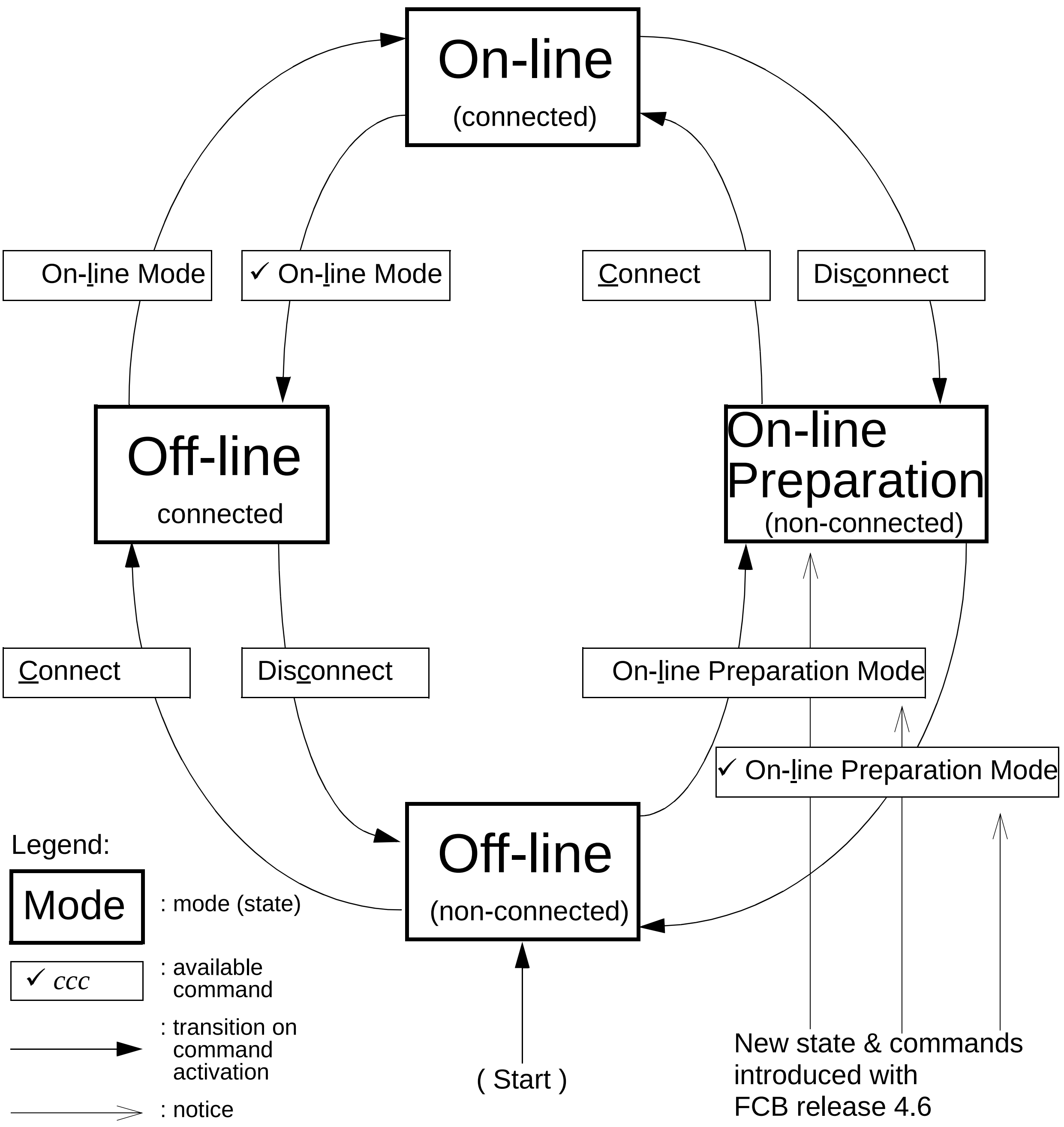


Figure 9-1. FCB Modes (States) and Transitions

Off-line (non-connected) Mode

The engineering station and the target system are not connected.
The application on the AC 100 Series target system and the corresponding node as seen by the Function Chart Builder may be different, they are not synchronized.

When editing in this mode, you can change any number of executable units.

The following commands are available:

- **Target | Connect** to switch to off-line connected mode
- If the difference between the application on the target and the node is not more than one executable unit also **File | On-line Preparation Mode** to switch to on-line preparation mode, in fact, to limit off-line editing to 1 executable unit.

The **Connect** command in the **Target** menu and the dimmed **On-Line Preparation Mode** command in the **File** menu or the missing check mark (✓) before this command indicate this mode.

NOTES

You work with the Function Chart Builder in this mode after start-up.

In this state you can use the **Target | Connect** command also to connect any *other* AC 100 Series target system, for example, to identify it on the basis of its **Status Report** and **Options Report** or to diagnose it.

Off-line Connected Mode

The engineering station and the target system are connected.
The application on the AC 100 Series target system and the corresponding node as seen by the Function Chart Builder may be different, they may be not synchronized.

When editing in this mode, you can change any number of executable units.
You download the target code of the whole application generally in this mode.

The following commands are available:

- **Target | Disconnect** to switch to off-line (non-connected) mode
- If there is a difference between the application on the target and the node, the **Target | Load Application** can be used to download the target code and switch to on-line mode immediately thereafter.
- If there is no difference between the application on the target and the node, also **File | On-line Mode** to switch to on-line mode and edit and test the application on the target on-line.

The **Disconnect** command in the **Target** menu and the dimmed **On-Line Preparation Mode** command in the **File** menu or the missing check mark (✓) before the **On-Line Mode** command in the **File** menu indicate this mode.

On-line Mode

The engineering station and the target system are on-line connected.
The application on the AC 100 Series target system and the corresponding node as seen by the Function Chart Builder are the same, they are synchronized.

In this mode, you can test and edit the application on the target on-line.
When editing in this mode, you can change at most one executable unit.

The following commands are available:

- **Target | Disconnect** to switch to on-line preparation mode, in fact, to preserve the limitation to one executable unit also during off-line editing
- **File | On-line Mode** to switch to off-line connected mode, in fact, to allow off-line editing beyond the one executable unit limit.

The check mark ` ✓ ´before the **On-Line Mode** command in the **File** menu indicates this mode.

On-line Preparation Mode

The engineering station and the target system are not connected.
The application on the AC 100 Series target system differs from the corresponding node as seen by the Function Chart Builder by not more than one executable unit.

When editing in this mode, you can change at most one executable unit.

The following commands are available:

- **Target | Connect** to switch to on-line mode
- **File | On-line Preparation Mode** to switch to off-line (non-connected) mode, in fact, to allow off-line editing beyond the one executable unit limit.

The check mark ` ✓ ´before the **On-Line Preparation Mode** command in the **File** menu indicates this mode.

NOTES

To reach this mode, the Function Chart Builder has to have passed the on-line mode at least once with this application.

In this state you can use the **Target | Connect** command also to connect any *other* target system, for example, to identify it on the basis of its **Status Report** and **Options Report** or to diagnose it. Then also the **Target | Disconnect** command is available. The on-line preparation mode regains its meaning only when you reconnect the *same* target system, with which you worked in on-line mode.

Transitions

Table 9-1. FCB Mode (State) Transitions

Transition	Command to Trigger the Transition
Off-line (non-connected) Mode ---> Off-line Connected Mode	Target | Connect
Off-line Connected Mode ---> Off-line (non-connected) Mode	Target | Disconnect
Off-line Connected Mode ---> On-line Mode	File | On-line Mode or Target | Load Application
On-line Mode ---> Off-line Connected Mode	File | On-line Mode
On-line Mode ---> On-line Preparation Mode	Target | Disconnect
On-line Preparation Mode ---> On-line Mode	Target | Connect concerning the *same* target
On-line Preparation Mode ---> Off-line (non-connected) Mode	File | On-line Preparation Mode
Off-line (non-connected) Mode ---> On-line Preparation Mode	File | On-line Preparation Mode

9.2 Switching to On-line Mode

On-line configuration and programming is the modification of a user application program on an engineering station while the engineering station is connected to an Advant Controller 100 Series system.

Modification in this case includes insertion and deletion of executable units or DB elements. An executable unit contains a structure element (with cycle time) and subordinate PC elements.

On-line configuration and programming is only possible by means of the Function Chart Builder running on an engineering station in *on-line mode*.

DANGER

Process equipment may be damaged!

Consequences of on-line modifications must be fully considered and understood before taking actions.

To switch from off-line connected mode to on-line mode with the Function Chart Builder, proceed as follows:

1. Check the options and the base software in the **Options Report** window for conditions of successful on-line operations:
   a. The base software and all the options selected in the Advant Station 100 Series Engineering Station must be available in the on-line connected target.
   b. The *ident_string*, *major_release*, and *minor_release* of the item selected in the Advant Station 100 Series Engineering Station must equal the *ident_string*, *major_release*, and *minor_release* of the corresponding item on the target. That means *revision* is not evaluated.
2. Check that the data in the **Node on AS 100ES** column of the **Status Report** window equal those in the **Target RAM** column.
   Otherwise reload the application. See Section 8.2, Load Application.
3. Set on-line mode by the **File | On-line Mode** command.

The Function Chart Builder offers switching from off-line connected mode to on-line mode also after the successful execution of the **Target | Load Application** command.

To switch from on-line preparation mode to on-line mode with the Function Chart Builder, proceed as follows:

Choose the **Target | Connect** command.

9.3 Modifying the Application Program On-line

To edit the application program on-line mode with Function Chart Builder, proceed as follows:

1. Update the **Status Report** by the **Target | Report Status** command.
   Check **PCPGM State** in the **Status Report**.
   Be aware whether the PC program is blocked or deblocked.
2. Select the **PC Section** window or the **DB Section** window and edit an executable unit on-line in accordance with the on-line editing rules in Section 9.4, On-line Editing Rules.
3. After leaving an editor with the **File | Save and Load** command, the modified executable unit or the modified DB element is converted to target code, downloaded to the AC 100 Series target, and switched to execution automatically.

An on-line modified user application program is downloaded to the target system in parts of a complete user application program. The system divides these parts automatically.

Special handling of sequences:

> During on-line programming applications with sequences, if you insert a new STEP element or delete an old STEP element, the sequence is first executed once in RESET-mode after on-line change.

Suggested Next Steps

1. You can also monitor and force terminals in the target. See Chapter 10, Test Functions.
2. You can upload the new parameter values from the target.
3. Or just switch off-line and disconnect the target by the **Target | Disconnect** command.

9.4 On-line Editing Rules

In contrast to off-line editing, restricted handling is possible when the Function Chart Builder is in on-line mode or on-line preparation mode, that is, you make on-line changes in a running application or prepare them:

- Changes such as inserting, modifying, and deleting PC elements are allowed for only one executable unit at a time.
- Modification of the scheduling strategy of a control module (cyclic activation, activation at initialization, and so on) is not allowed.
- The first change fixes the executable unit.
- Deleting connections is always allowed in the fixed executable unit and between the fixed executable unit and one other executable unit.
- Connections to constants and to DB elements are only allowed in the fixed executable unit.
- Adding connections between executable units is only allowed in the following way:
  a. Fix the executable unit having the input terminal.
  b. Connect the input terminal of the fixed executable unit to the output terminal of the other executable unit.

- Some terminals of DB elements cannot be changed on-line.
- The PC and DB sections cannot be changed simultaneously.
- Since Function Chart Builder 4.7 also the on-line creation and deletion of simple type circuits is supported. These are either DB-only type circuits or PC-only type circuits containing no executable units.

NOTE

All these restrictions hold in the case of *substantial changes*, that is, for those that are reflected by the target code and are conveyed to target.

Changes of DB element names, PC element names, and PC signal names, item designations, page divisions, and header and footer texts are generally *unsubstantial*.

Though, in case of AC 100 Series nodes supporting application retrieval or if you generate also information for HMI, you may have completed the target code with some or all of these items. Then also their changes become substantial.

You must use the **File | Save and Load** command to store the changes on the AC 100 Series target. Thereafter, you can modify some other executable unit.

For detailed description of windows, commands, and dialogs see the *Function Chart Builder User's Guide*.

Chapter 10 Test Functions

10.1 General

The test functions perform value displaying (monitoring), value forcing, and transitory value setting. These test functions are available in the Function Chart Builder running on the engineering station in *on-line mode* only.

The Function Chart Builder is in on-line mode if the following conditions are fulfilled:

1. The related Advant Controller 100 Series target is connected.
2. The same version of the application program is selected on both the engineering station and the AC 100 Series target.
3. You switched on-line by the **File | On-line Mode**, **Target | Connect**, or **Target | Load Application** command.

Details see in Section 9.2, Switching to On-line Mode.

The test values are presented by Function Chart Builder in one or several **Test Table** windows. The PC test values are presented also in **PC-Section** window panes with the function chart

NOTES

Since FCB version 4.4/0, the **Test Table** replaces the earlier **PC Display Table** window, **PC Force Table** window, and **DB Force/Display Table** window.

Value display (monitoring) and forcing of DB terminals are parts of the test functions of the engineering station since FCB version 4.3/0.

The selection of variables is independent of the activation of displaying or forcing of these variables on the AC 100 Series target.

With the command **Window | Test Table** you create or open the **Test Table** window.
To overview and check the values of interest, you can resize the columns by dragging the divisions that separate the header parts **Instance**, **Terminal**, and so on. You can change the order of the columns by dragging the header parts to the required position. You can set or invert the sort order by clicking on the header part, according to which you want to sort the entries.

10.2 Configuration Part

Select the DB terminals for monitoring or forcing by activating the **Create/Edit…** button in the **List Entry** group of the **Test Table**. You can do this both in on-line and off-line modes.

Using the commands of the **Test** menu, you can activate the tests on the AC 100 Series target.

Transitory DB terminal value setting is done with own **Test** command. You enter the value in the **Set DB Value** dialog box each time you want to set a DB terminal in a transitory way. This value is not contained or stored together with the force values.

Section 11.9.1, Configuration Part describes the execution of display, force, and set DB element values in the controllers.

10.2.1 Value Display (Monitoring) of DB Terminals

Terminals to Be Monitored

You can select the following DB terminals for display:

DB terminals connectable to PC elements.

Display Values

- An asterisk ‘*****’ is displayed if the value not known yet in the engineering station (for example, no **Display** command was executed).
- ‘**x**’ is displayed if the internal representation of a Boolean value is not hexadecimal 00FF (= FALSE) or FF00 (= TRUE). This can occur, for example, if a Boolean variable has never been written before and the data area was initialized with zeros (0000).
- “Not convertible” is displayed if a value is not convertible for displaying (for example, out of range).
- “No conversion (NAN)” is displayed if real variable has a value that is “Not A Number” (NAN). NAN is defined as invalid value for I/O modules.

The Function Chart Builder stores the marking of terminals to be monitored on the engineering station and restores them at the beginning of the next session.

You can use appropriate **Test** commands to export or import the **Test Table** to or from a file, respectively, or upload force values from the controller to the **Test Table**.

10.2.2 Forcing DB Terminals

CAUTION

Process equipment may be damaged!

Consequences of forcing values must be fully
considered and understood before taking actions.

Be aware that force values on the target are valid NOT only
during being in the test function of the engineering station.
When you leave the test functions on the engineering station,
all the force values on the target REMAIN effective.

Terminals to Be Forced

You can select the following terminals for forcing:

VALUE terminal of extended DB elements.

Force Values

On activation the force values are downloaded to the target system and linked to the data base.

Loaded force values become active on the target system at once, or at the latest, after the cycle time of the involved DB elements.

Forcing DB terminals is not a temporary function. When you leave the test functions on the engineering station, all the force values remain set on the target.

The Function Chart Builder stores the marking of terminals to be forced and their values on the engineering station and they are available at the beginning of the next session.

10.2.3 Setting DB Element Terminals

Terminals to Be Set

You can select the following DB terminals for transitory setting:

- DB terminals connectable to PC elements
- VALUE terminal of extended DB elements.

Set Values

On activation the set values are downloaded to the target system and linked temporarily to the data base.

Loaded set values become active on the target system at once and are valid until changed by any source (task) independently from whether you leave the test functions on the engineering station or not. A common use of this function is the following:

1. Block the application program in the controller by **Target | Block Program** command.
2. Set some DB terminal values. These values become effective.
3. Deblock the application program in the controller by **Target | Deblock Program** command. The DB element terminal values will be changed by the application program in the controller at once, or at the latest, after the cycle time of the involved DB and/or PC elements.

The Function Chart Builder does not store the marking of terminals to be set nor their values on the engineering station, they are not available at the beginning of the next session.

10.3 Program Part

Value display (monitoring) and forcing of PC terminals are parts of the test functions of the engineering station.

The Function Chart Builder presents the monitored and forced values in the following ways:

- Function chart representation giving information about the states of local sections within an application
- Table representation collecting the variables distributed over the application.

The selection of variables is independent of the activation of displaying or forcing of these variables on the AC 100 Series target.

With the commands **PC-Terminal | Select for Display** and **PC-Terminal | Force Value** you select and mark the terminals you want to test. Using the commands of the **Test** menu, you can activate the tests on the AC 100 Series target.

Section 11.9.2, Program Part describes the execution of display and force PC element values.

10.3.1 Value Display (Monitoring)

Terminals to Be Monitored

You can select the following terminals for display:

- Terminals connected to other PC elements within the same executable unit
- Terminals connected to other executable units (marked with ‘P’ or ‘G’)
- Terminals connected to DB elements (marked with ‘P’ or ‘G’)
- Unconnected output terminals.

You cannot select the following terminals for display:

- Unconnected input terminals
- Input terminals connected to constants
- Terminals which are treated as comment on AC 100 Series targets (for example, MMC functions in SEQ/STEP)
- Terminals within SLAVE modules not belonging to a MASTER module.

Display Values

- An asterisk ‘*****’ is displayed if the value was not determined by the AC 100 Series target (for example, PC program has never been executed).
- ‘**x**’ is displayed if the internal representation of a Boolean value is not hexadecimal 00FF (= FALSE) or FF00 (= TRUE). This can occur, for example, if a Boolean variable has never been written before and the data area was initialized with zeros (0000).
- “Not convertible” is displayed if a value is not convertible for displaying (for example, out of range).
- “No conversion (NAN)” is displayed if real variable has a value that is “Not A Number” (NAN). NAN is defined as invalid value for I/O modules.

The Function Chart Builder stores the marking of terminals to be monitored on the engineering station and restores them at the beginning of the next session.

10.3.2 Forcing PC Terminals

CAUTION

Process equipment may be damaged!

Consequences of forcing values must be fully considered and understood before taking actions.

Be aware that force values on the target are valid ONLY during being in the test function of the engineering station. When you leave the test functions on the engineering station, all the force values on the target will be CLEARED.

Terminals to Be Forced

You can select the following terminals for forcing:

- Terminals connected to constants
- Terminals connected to DB elements (marked with ‘P’ or ‘G’)
- Terminals connected to other executable units (marked with ‘P’ or ‘G’).

You cannot select the following terminals for forcing:

- Unconnected input terminals
- Terminals connected to other PC elements within the same executable unit only
- Terminals which are treated as comment on AC 100 Series target (for example, MMC functions in SEQ/STEP)
- Terminals within SLAVE modules not belonging to a MASTER module.

Force Values

The force values are downloaded to the target system and linked to the related executable unit.

Loaded force values become active on the target system at the next execution of the related executable unit.

Forcing terminals is a temporary function. When you leave the test functions on the engineering station, all the force values on the target will be cleared.

Normally, terminals that cannot be forced, cannot be selected for forcing, either. Though, in some cases they may be selected. Then the **Test | Activate Force Values** command generates an error message on the engineering station and **clears all** force values after confirmation.

The Function Chart Builder stores the marking of terminals to be forced and their values on the engineering station and they are available at the beginning of the next session.

NOTE

> As shown in Section 11.8.2, Program Part, the values displayed on the engineering station are determined at the **end of the execution** of the related control module.

Chapter 11 Execution

11.1 Internal Software of Advant Controller 100 Series

The internal software of the AC 100 Series systems is schematically illustrated below.

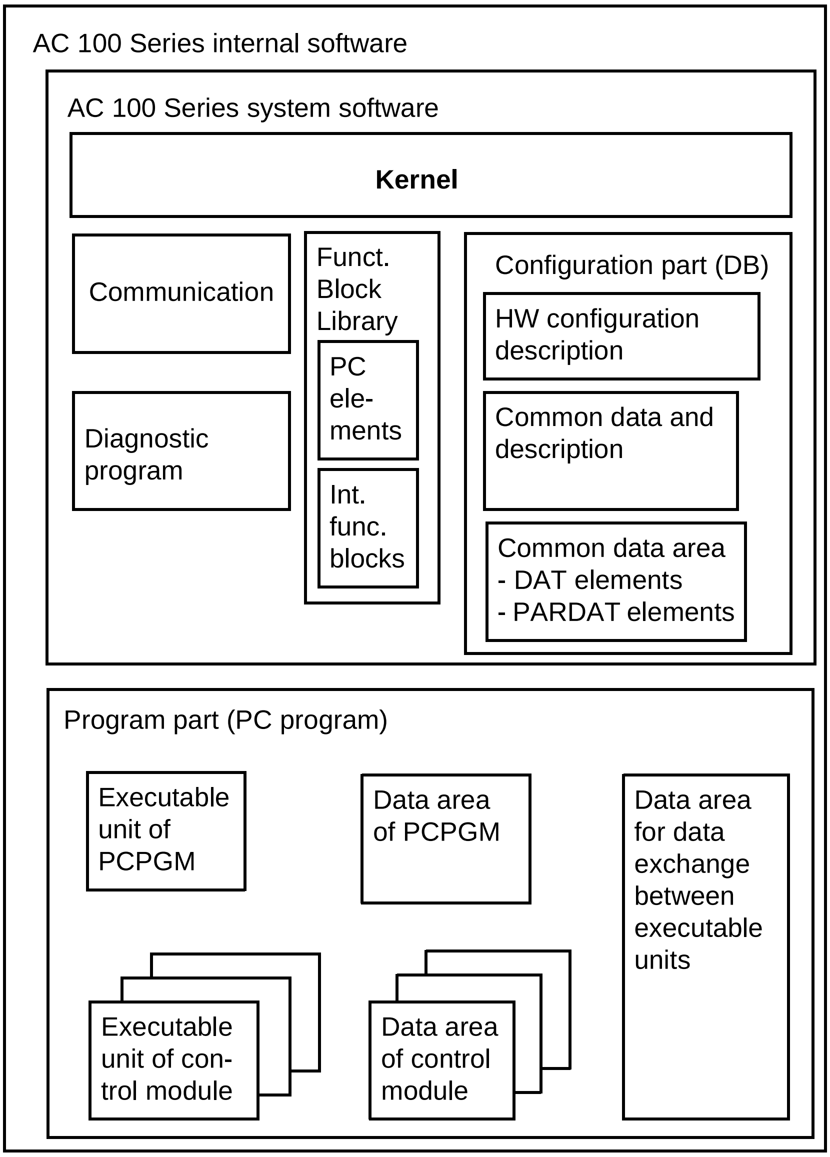

AC 100 Series internal software
AC 100 Series system software
Kernel
Communication
Funct. Block Library
PC ele-ments
Int. func. blocks
Configuration part (DB)
HW configuration description
Common data and description
Common data area
- DAT elements
- PARDAT elements
Diagnostic program
Program part (PC program)
Executable unit of PCPGM
Data area of PCPGM
Data area for data exchange between executable units
Executable unit of con-trol module
Data area of control module

Figure 11-1. Block Diagram of AC 100 Series100 Series Internal Software

11.1.1 Kernel

The PC program and the related control modules (the executable units) in an AC 100 Series system must coexist with other types of programs such as diagnostic and communication programs. One task of the kernel is to schedule the execution of the different programs.

For the PCPGM and the control modules, the priority for scheduling is given by the cyclicity and the place in the cycle time table. This is done by the call parameters of the elements PCPGM, CONTRM, MASTER, and SEQ.

The kernel schedules, at each point in time, the execution of the program with the highest priority. This may have the effect that the execution of an execution unit is interrupted temporarily by another executable unit. This, however, has no effect on the function of the PC program.

11.1.2 Communication

The communication part of the AC 100 Series internal software supports the communication with the tools.

The communication must perform the following tasks:

- Organization of data transfer via Advant Fieldbus 100 and/or RCOM and transfer of data
- Services transferred via the communication (for example, the on-line change of an executable unit or read and delete the error buffer).

11.1.3 Diagnostic Program

The diagnostic program performs diagnosis of the CPU and the connected modules.
The diagnostic program is running in the background, when no other program part needs to be executed.

The diagnostic program writes checked error information into an error buffer. The error buffer can be read and deleted by means of an engineering station via the communication part of the AC 100 Series internal software.

11.1.4 Function Block Library

The function block library contains the execution code of the following items:

- PC elements
- Internal function blocks, such as
  functions for display and force of values and
  GET and PUT move blocks between data areas of PCPGM and of control modules on the one side, and I/O and communication modules and channels, common data area, data area for data exchange between executable units on the other side.

11.2 Process I/O Handling

AC 100 Series systems exchange data with the process. Depending on controller type, this is performed directly via backplane-connected I/O units or the process I/O is built in the basic unit and I/O expansion units.

Data for these units are stored in the following different data areas:

- Hardware configuration description area containing configuration data for the I/O modules and channels
- Local data areas of PCPGM and of control modules containing process data.

Data transfer of configuration data to the I/O is performed during system initialization and for on-line exchanges after the download of new configuration data.

Data exchange between the process and the local data areas is performed directly by the related executable units within their input and output phases.

11.3 Input and Output Phases

To isolate each execution unit from data change during execution, the execution is commenced by an input phase and ends with an output phase.

This data exchange is performed by GET elements for the input phase and PUT elements for the output phase.

The PUT and GET elements are internal function blocks of AC 100 Series.

PUT and GET elements are automatically inserted by the engineering stations.
They are indicated in the PC diagram by a ‘P’ or a ‘G’.

11.3.1 Input

During the input phase of an executable unit, the used values are stored in the local data area of the executable unit. Sources of these data are the following:

- I/O and communication modules and channels
- Common data area
- Data area for data exchange between executable units.

The necessity of having static data during the entire execution of a unit is illustrated by Figure 11-2.

Suppose an internal interrupt occurred between the execution of elements 20 and 21 due to value change 1 ---> 0 on the input ON.

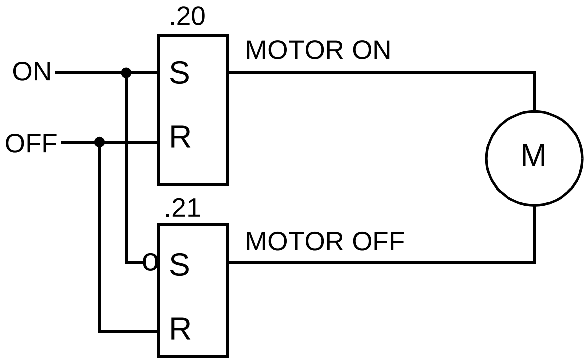


Figure 11-2. Example for the Necessity of Having Input Elements

If the data were not static during the entire execution of the unit, the motor would be ordered on and off simultaneously.

11.3.2 Output

During the output phase, the calculated data from the elements in an execution unit are transferred to the following sinks:

- I/O and communication modules and channels
- Common data area
- Data area for data exchange between executable units.

The data is fetched directly from the storage location of each element output.

11.4 Execution Order

AC 100 Series systems processes the different execution units in the following order:

- Processing of PC program (processing of PCPGM)
  1. Reading input values of the structure element PCPGM
  2. Processing structure element PCPGM
  3. Writing output values of the structure element PCPGM
  4. For AC 110, 160, and 70 also transfer of common data between common data area and Advant Fieldbus 100 DataSet communication.
- Processing of control module CONTRM
  1. Reading input values of the structure element CONTRM
  2. Processing structure element CONTRM
  3. Writing output values of the structure element CONTRM
  4. Reading input values in the control module
  5. Processing elements in the control module
  6. Writing output values in the control module.

- Processing of control module MASTER

  1. Reading input values of the structure element MASTER
  2. Processing structure element MASTER
  3. Writing output values of the structure element MASTER
  4. Reading input values of the first slave module
  5. Reading input values of the second slave module
  .
  .
  n + 3 Reading input values of the n-th slave module
  n + 4 Processing elements of the first slave module
  n + 5 Processing elements of the second slave module
  .
  .
  2n + 3 Processing elements of the n-th slave module
  2n + 4 Writing output values of the first slave module
  2n + 5 Writing output values of the second slave module
  .
  .
  3n + 3 Writing output values of the n-th slave module.

- Processing of control module SEQ

  1. Reading input values of the structure element SEQ
  2. Reading input values of the first structure element STEP
  3. Reading input values of the second structure element STEP
  .
  .
  .
  n + 1 Reading input values of the n-th structure element STEP
  n + 2 Processing control part of structure element SEQ
  n + 3 Writing output values RUN and MODP of the structure element SEQ
  n + 4 Processing sequence part of the structure element SEQ
  n + 5 Reading input values of the active STEP module
  n + 6 Processing elements of the active STEP module
  n + 7 Writing output values of the active STEP module
  n + 8 Writing output values except RUN and MODP of the structure element SEQ.

On step change, the active step execution is repeated and the new step is then executed. This is always performed for the entire step with its input and output elements.

11.5 Execution of Elements

Execution of the elements commences after the execution of the header of an execution unit and the execution of the input elements.

The PC element itself controls its execution by calling the code in the element library for the next element. This is done in the order of storage, which also corresponds to the entry and documentation order.

The program code in the library works with a part of the PC program corresponding to a single element. The internal working principle is shown below for an ADD with two inputs.

The ADD part is used by the preceding element to select the appropriate code from the library.

This program code reads data from the data area at addresses *a* and *b* and calculates the sum of this data. The result is then stored at address *c*. Then the ADD program code reads the type-code of the next element and calls the related element code to proceed with the next element in the PC program.

PC elements coded with in-line code are an exception. These elements are coded directly as processor instructions.

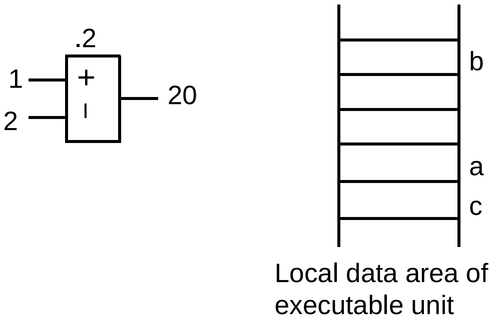


Figure 11-3. Execution of an ADD Element

11.6 Reset Execution

Clearing or reset execution takes place when an R input of a module header is active. This has the effect of setting all variables in the local data area of the module to an initial value. For most variables, this implies that they are set to the zero value of their respective data type. There are, however, certain exceptions to this, since the outputs of a certain element must have consistent values, see Figure 11-4.

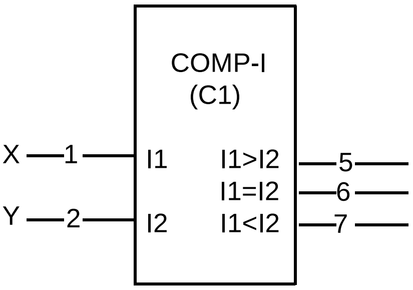


Figure 11-4. Comparator COMP-I

Since X and Y are both 0 with reset execution, the output I1=I2 must be 1 and the other outputs must be 0 in order for the outputs to have consistent values.

Following reset execution, the output phase is performed with the values generated in the local data area.

Reset execution for the different execution units when R=1:

• **CONTRM**	Reset is executed for all elements in each cycle.
• **MASTER**	Reset is executed for all elements of all slaves in each cycle.
• **SEQ**	Reset is executed for all elements of all steps in one cycle when R goes to 1.
• **PCPGM**	Reset is executed for all constituent execution units as stated above.

11.7 Initialization

Following a cold start, defined values are assigned to the data in an AC 100 Series system. This is performed by executing all modules and elements in init mode. For the elements, this is normally equivalent to reset execution. Possible exceptions to this are stated in the element data sheets. After warm-start, the AC 100 Series keeps its values and continues the flow of the program as if it was interrupted by power-down.

11.8 On-line Changes

11.8.1 Configuration Part

On-line changes in the configuration part (see Chapter 9, On-line Configuration and Programming) are executed on the AC 100 Series target in the following way:

- Insertion of a new DB element
  - Module or channel element
    The new element will be initialized and configured as during a cold start of the AC 100 Series system (see *Advant Controller 110 User's Guide*, *Advant Controller 160 User's Guide*, *Advant Controller 70 User's Guide*, *Advant Controller 55 User's Guide*).
  - Common data element
    - A new common data area will be generated.
    - The actual values of the old common data area are transferred to the new common data area.
- Modification of an existing DB element
  - Module or channel element
    The modified module and its related channels will be initialized and configured as during a cold start of the AC 100 Series system (see *Advant Controller 110 User's Guide*, *Advant Controller 160 User's Guide*, *Advant Controller 70 User's Guide*, *Advant Controller 55 User's Guide*).
  - Common data element
    The only possibility to modify a common data element is the modification of an initial value of a PARDAT element. This modification is valid at next cold start on the AC 100 Series system.

- Deletion of an existing DB element
  - Module or channel element
    - The complete module will stay in the actual state.
    - In case of an AC 110, 160, or 55 system, the involved module remains in its actual state but it will not be initialized and configured after next cold start.
    (See *Advant Controller 110 User's Guide*, *Advant Controller 160 User's Guide*, *Advant Controller 55 User's Guide*.)
    - In case of an AC 70 system, the involved module will be set to "not configured" state at once. (See *Advant Controller 70 User's Guide*.)
  - Common data element
    - The complete element will stay in the actual state.
    - After next cold start of an AC 100 Series system the deleted element will not be initialized and configured.

11.8.2 Program Part

On-line changes in the program part (see Chapter 9, On-line Configuration and Programming) are executed on the AC 100 Series target in the following way:

- Insertion of new executable unit

  The new executable unit will be installed and started.
- Modification of an existing executable unit
  - If there are other executable units with new PUT elements to generate data for the actual executable unit:
    - The executable units with new PUT elements will be installed.
    - A bumpless switch-over is performed from the old executable unit to the new one.
  - The new (modified) executable unit will be installed.
  - A switch-over which is as far as possible bumpless is performed from the old executable unit to the new one.

NOTE

As far as possible bumpless means the actual data values of the old executable unit are copied during switch-over to the local data area of the new executable unit.

- Deletion of an existing executable unit
  - The executable unit will be stopped and deleted.
  - Deletion of the PC element PCPGM is not allowed.

In the case of inserting or modifying an executable unit, the data area for data exchange between executable units is also bumpless changed to an adapted new one.

11.9 Test

11.9.1 Configuration Part

Display Values

To display values of DB element terminals connectable to PC elements, the communication of the AC 100 Series system executes the following tasks:

- Downloads suitable move elements from the connected engineering station.
- Executes the move elements to transfer the related data to a special local data area after execution of the executable units.
- Uploads the contents of this data area to the connected engineering station.

Force Values

To force VALUE terminal of extended DB elements, the communication of the AC 100 Series system executes the following tasks:

- Downloads suitable move elements which include the force values from the connected engineering station.
- Executes the move elements to transfer the force data to the local data area during execution of the executable units.

  (The force values are moved after execution of the GET elements of the executable unit and before execution of the PUT elements of the executable unit.)

Set Values

To set DB terminals connectable to PC elements or VALUE terminal of extended DB elements in a transitory way, the communication of the AC 100 Series system executes the following tasks:

- Downloads suitable move elements which include the set values from the connected engineering station.
- Executes the move elements to transfer the set data to the local data area at once independently from the execution of the executable units.

  (The set values are moved before execution of the GET and PUT elements of the executable unit.)

11.9.2 Program Part

Display Values

To display values of PC elements, the communication of the AC 100 Series system executes the following tasks:

- Downloads suitable move elements from the connected engineering station.
- Executes the move elements to transfer the related data to a special local data area after execution of the executable units.
- Uploads the contents of this data area to the connected engineering station.

Force Values

To force (set) values of PC elements, the communication of the AC 100 Series system executes the following tasks:

- Downloads suitable move elements which include the force values from the connected engineering station.
- Executes the move elements to transfer the force data to the local data area during execution of the executable units.

  (The input values of PC elements are moved after execution of the GET elements of the executable unit.
  The output values of PC elements are moved before execution of the PUT elements of the executable unit.)

Appendix A Application Examples

We give you two examples of the procedure to prepare a PC draft. Program entry, editing, and documentation generation is not shown. Refer to *Function Chart Builder User's Guide*.

A.1 Control of Lift

The task is to control a passenger lift between two floors. The lift is powered by a motor with a forward/reverse switch as shown in Figure A-1.

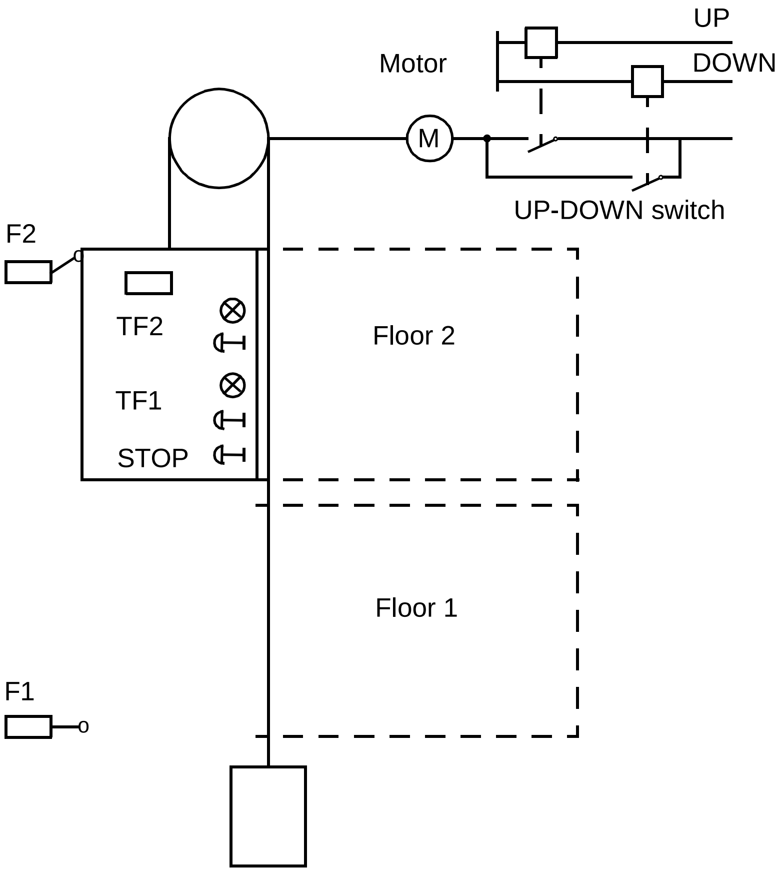


Figure A-1. Control of Lift

The lift is to be provided with supervision of overload and running time as well as supervision of the position of the door. The position of the lift carriage is detected with two position switches, one for each floor.

To check that the door is closed, another position switch is employed. To detect overload, the motor current is monitored.

The control equipment gives order to the forward/reverse switches for UP or DOWN.

One description of the process can be the block diagram in Figure A-2, which shows the external signals that affect the lift control and the signals used to control the lift motor and to indicate operational status.

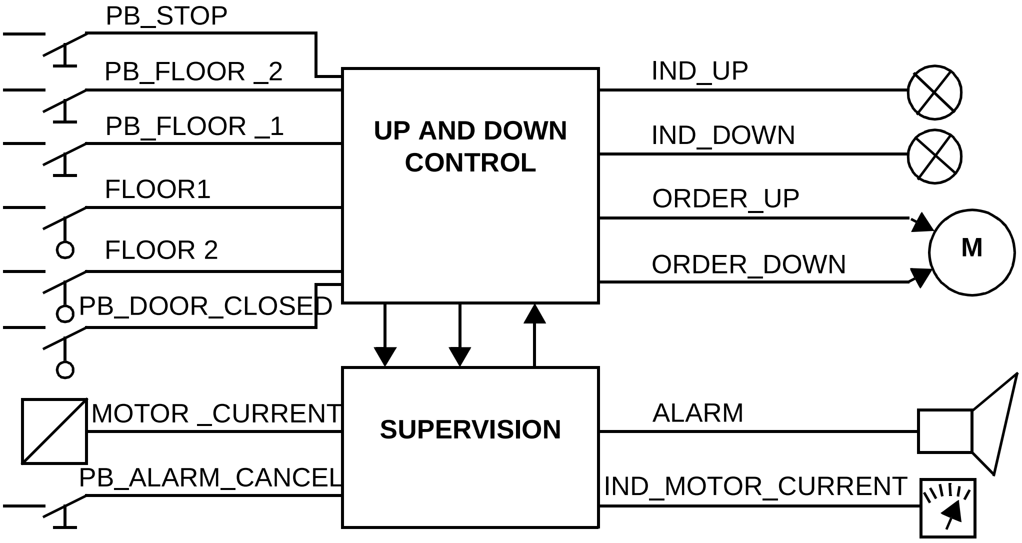


Figure A-2. Block Diagram of Process

The next operation is structuring the PC program, that is, dividing the program into PC modules and introducing module headers for execution control.

In this case, a division into control modules for control and supervision is necessary.

Having done this, a logic diagram for the control equipment can be drawn. This may appear as in Figure A-3. Two timer circuits have been used for supervision. If the running time exceeds four minutes, an alarm is triggered and the lift is stopped. Also, the lift is stopped and the alarm triggered if the motor current exceeds a certain value for more than 5 s.

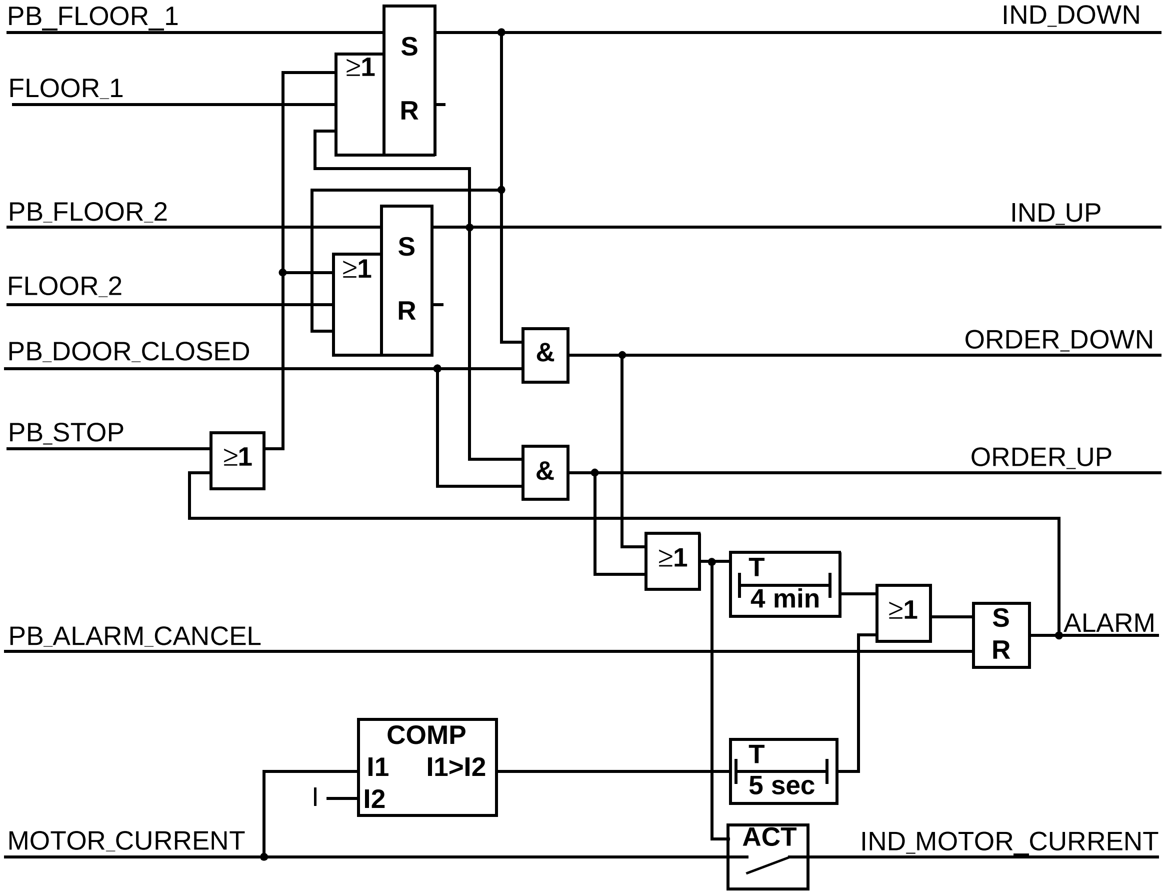


Figure A-3. Logic Diagram of Lift Control

This diagram specifies the desired function of the control equipment. It can now be converted into a draft PC diagram with elements chosen from the element library.

In this draft, the variable names and operational parameters (time values) for the delay elements are entered.

If you analyze the circuit from the execution point of view, you can see that the control part must be executed with a short execution cycle so that the stop signals can act fast enough.
The timing of the supervision part is not critical in the same way, for example, the delay time for overload alarm is set to 5 s. In Figure A-4, the circuit is divided into two PC modules, each with a control module header (CONTRM). The control part is to be given the cyclicity 20 ms and the supervision part 100 ms.

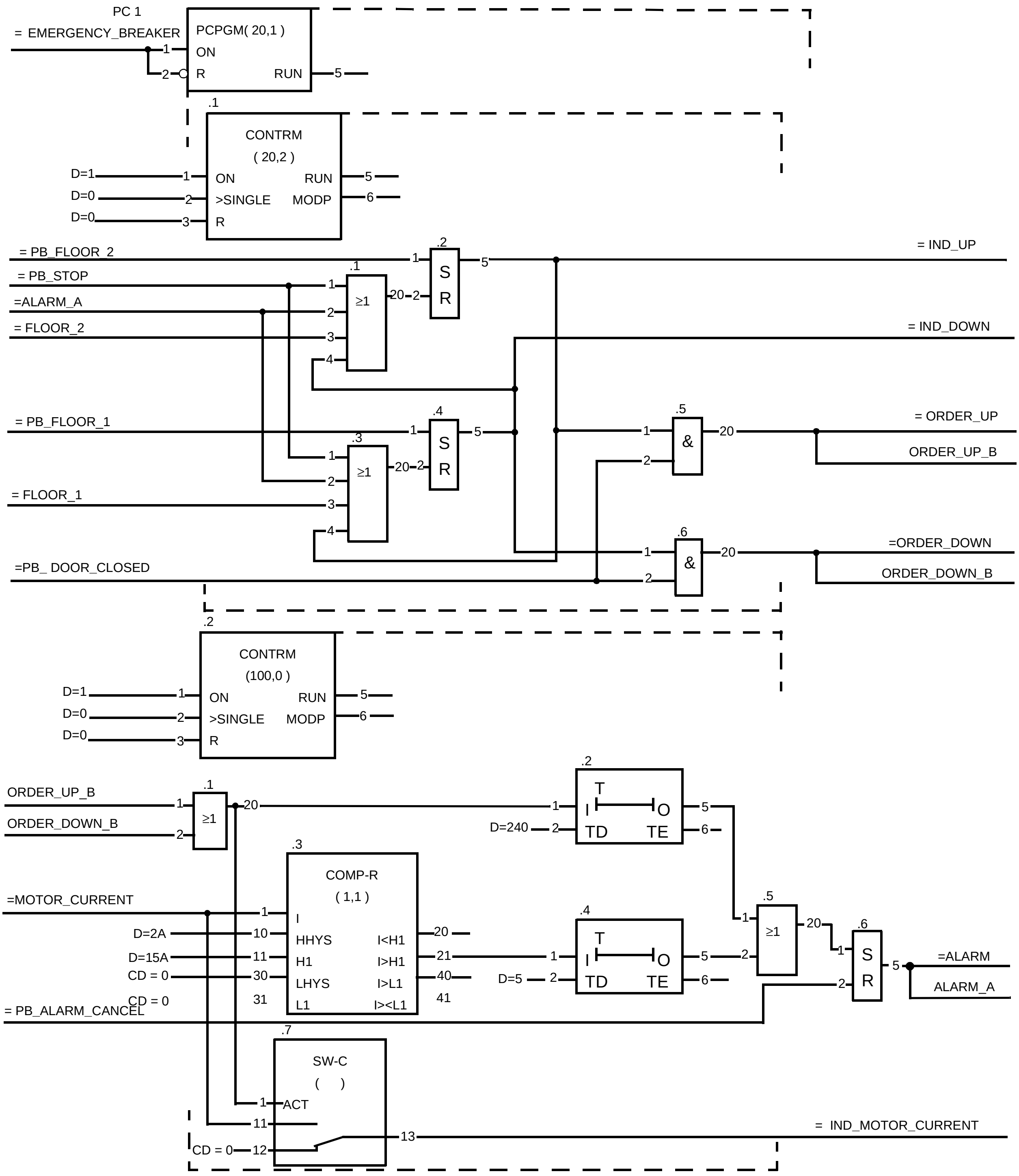


Figure A-4. PC Diagram with Two PC Modules, Draft

In the last step, the PC modules are provided with a superior PC program according to Figure A-5.

The PCPGM checks the emergency breaker. The PCPGM gets the same cycle time as the control part (20ms). To be executed before the control part, the PCPGM is given place 1 in the cycle time table and the control part is given place 2.

The cycle time of 100 ms is valid for one executable unit only. The supervision part is given place 0 (or any other) in the cycle time table.

Finally, item designations for all PC modules and PC elements are entered and the program is ready for entry.

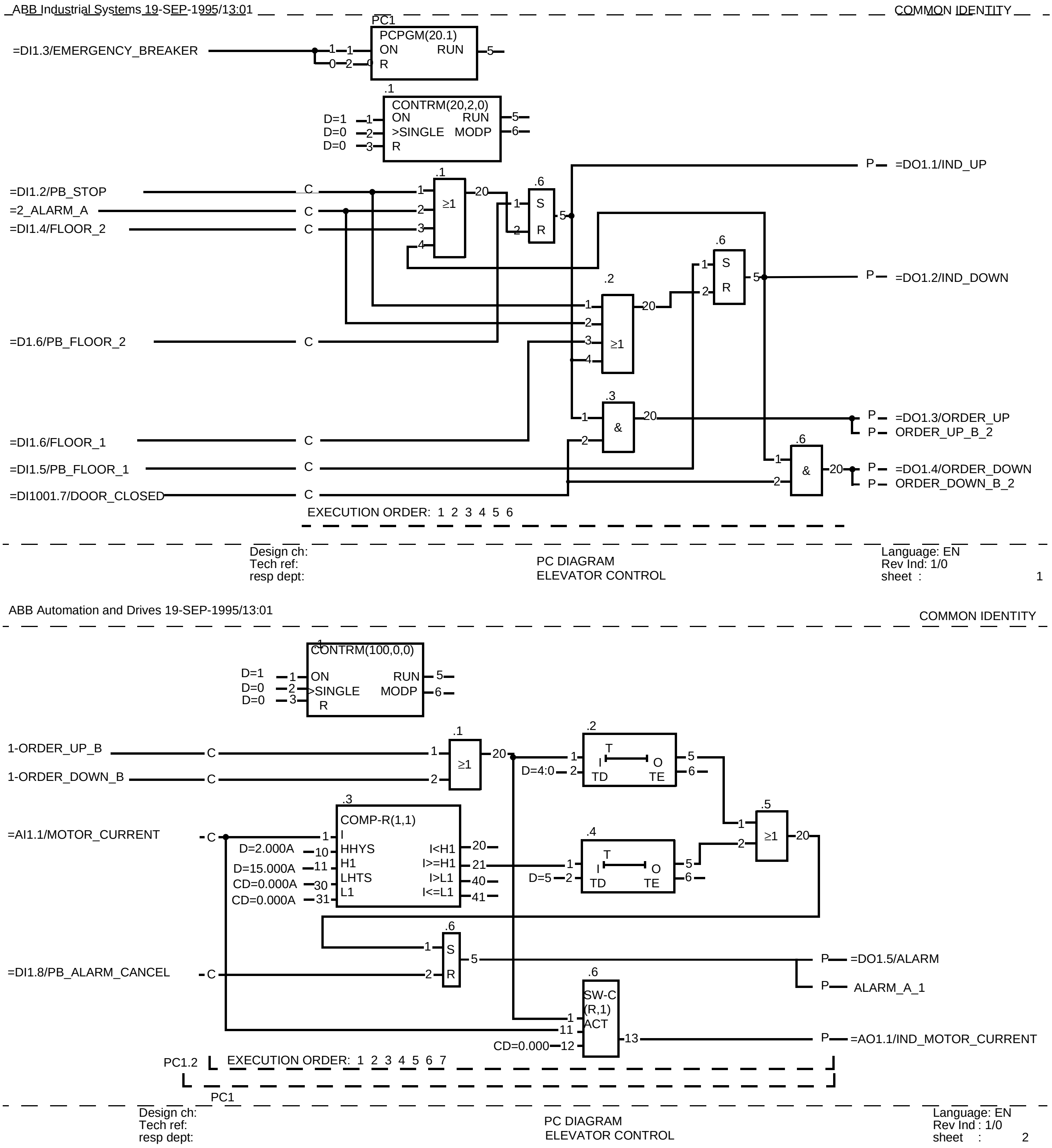


Figure A-5. Completed PC Diagram of Lift Control

A.2 Control of a Car Position

(Sequence Control)

Sequential control is the control of a process or machine performing a predetermined sequence of steps. This is illustrated by the following example in Figure A-6:

A car

1. Goes from its original position at A to position B
2. Stops
3. Waits
4. Returns to A.

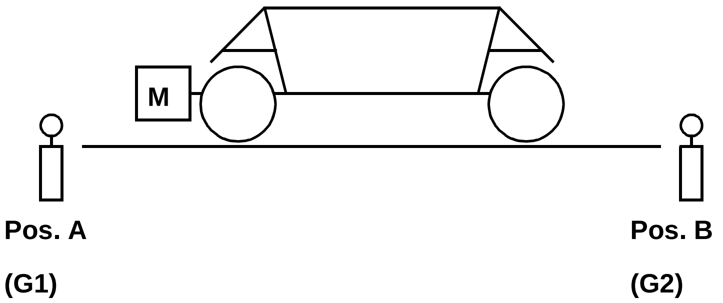
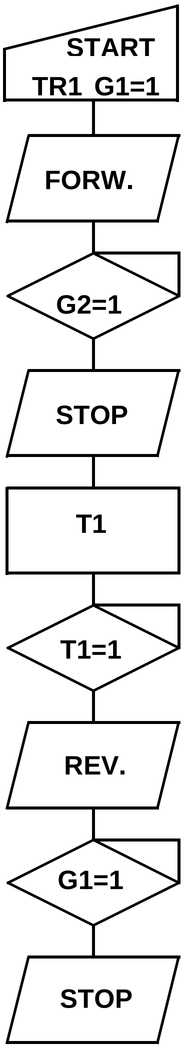


Figure A-6. Car Control

The sequence is described in a flow diagram as shown in Figure A-7.



Push button TR1 starts sequence if G1=1

The car runs FORWARD

Until G2=1

When it stops.

Timer circuit T1 is started.

When the time delay has elapsed

The car reverses

Until G1=1

Where it stops.

Figure A-7. Flow Diagram for Car Sequence Control

The sequence consists of a superior sequence header with the common functions for start, initializing, manual control, and so on, and a number of step modules, one for each step in the sequence. Each step module has its own step module header whose function is determined by a number of control signals.

A function diagram as shown in Figure A-8 often gives a clearer picture of a sequence. This manner of describing the sequence is in accordance with a draft IEC standard and is described in detail in Section 5.5.3, Application with Sequences. Always prepare such a diagram as a basis for the PC diagram.

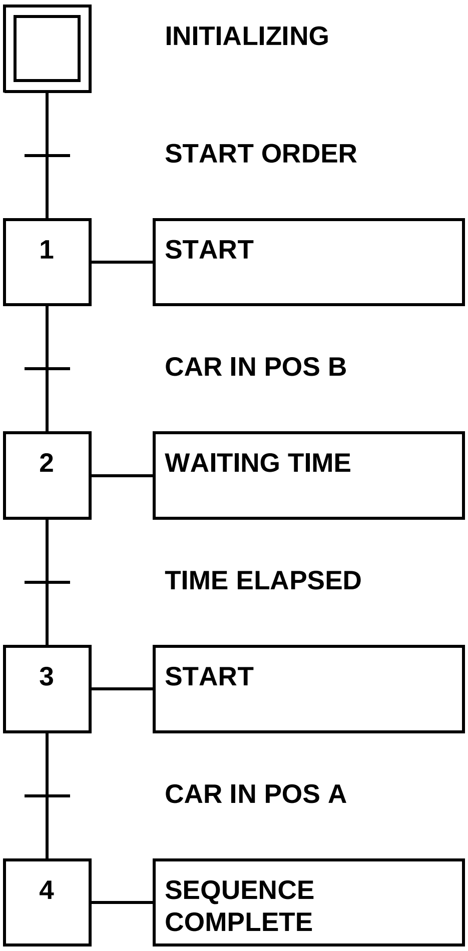


Figure A-8. Function Diagram for Car Control

Figure A-9 shows how the sequence is implemented in the Advant Controller 100 Series system.

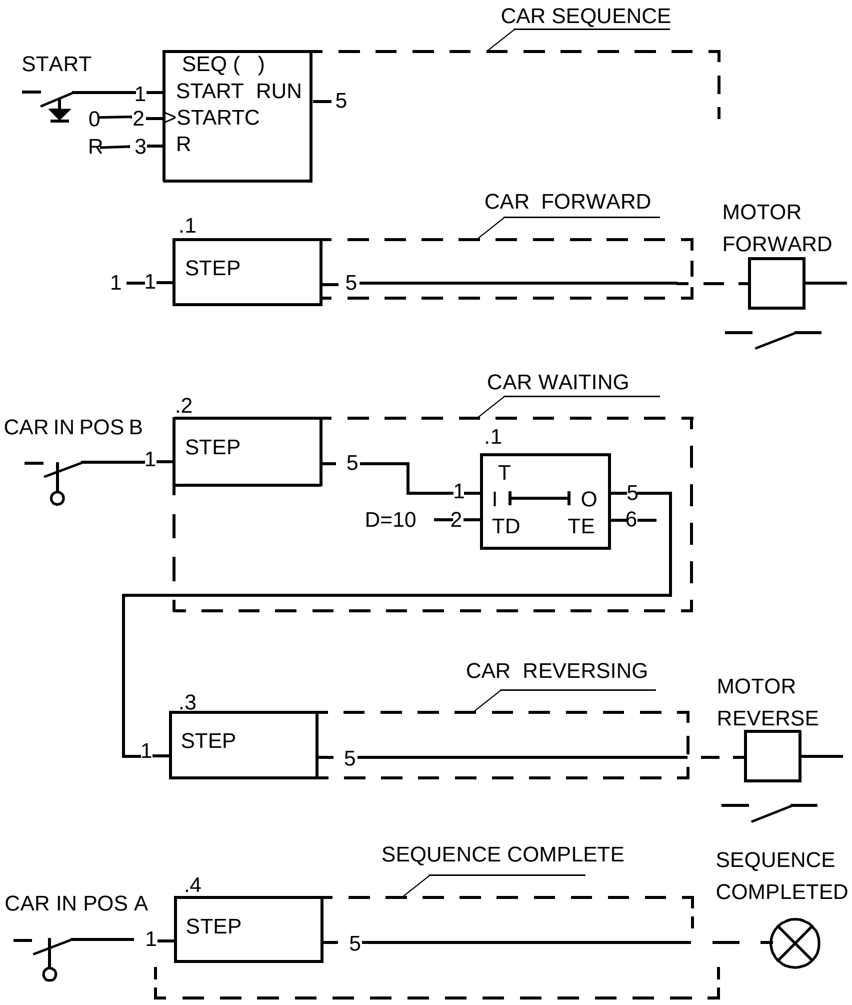


Figure A-9. Car Sequence Implemented in an AC 100 Series System

Appendix B Data Flow in Advant Station 100 Engineering Station

B.1 General

This appendix shows the internal data flow of the Advant Engineering components, that is, AMPL Control Configuration or AdvaBuild for Windows components, running on the Advant Station 100 Series Engineering Station:

The following Advant Engineering components are involved:

- Application Builder
- Function Chart Builder
- On-line Builder.

B.2 Files

*.ODB Files

*.ODB files contain internal data of the Function Chart Builder, which are specific for the objects in AdvaBuild for Windows (node, type circuit, circuit, and node structure).

*.C*nn* Files

*.C*nn* files contain target code for Advant Controller 100 Series applications.
They are internal data of the Function Chart Builder.

AMPL Source Code Files

For AC 100 Series applications, no AMPL source code is required.

AMPL source code (ANSI, 8 bit) may be used for the following tasks:

- Exchange application programs with other nodes (Advant Controller 100 Series, Advant Controller 400 Series, and MasterPiece 200/1 systems)
- Back up applications
- Modify applications by means of a text editor.

AMPL source code (ANSI, 8 bit) must be used for the following tasks:

- Decompose type circuits
- Upgrade Function Chart Builder version.

Derivatives of AMPL source code (ANSI, 8 bit) are used as source code for other objects than nodes.

B.3 Data Flow Diagram

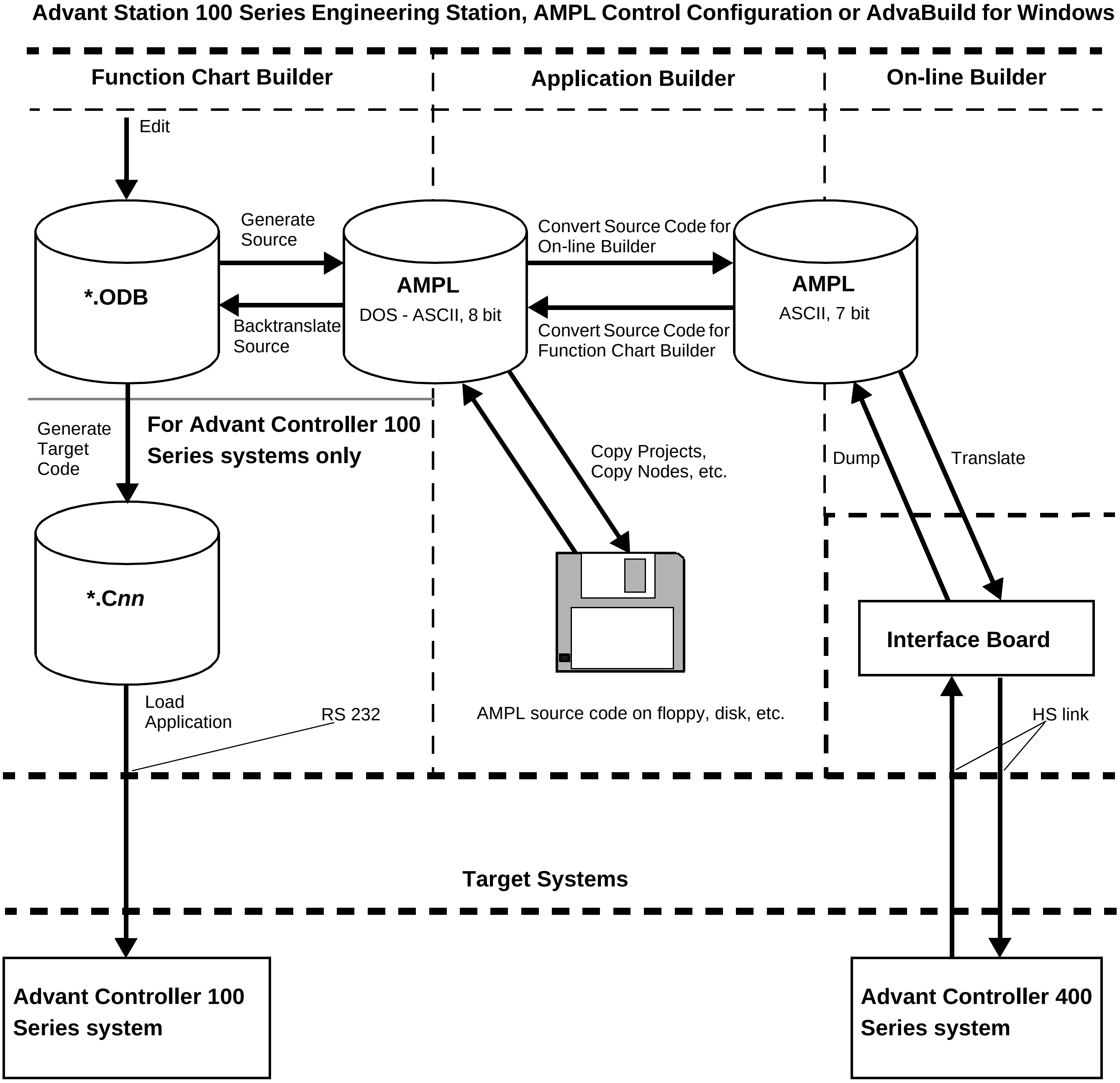

Advant Station 100 Series Engineering Station, AMPL Control Configuration or AdvaBuild for Windows
Function Chart Builder
Application Builder
On-line Builder
Edit
*.ODB
Generate Source
Backtranslate Source
AMPL
DOS - ASCII, 8 bit
Convert Source Code for On-line Builder
Convert Source Code for Function Chart Builder
AMPL
ASCII, 7 bit
Generate Target Code
For Advant Controller 100 Series systems only
Copy Projects, Copy Nodes, etc.
Dump
Translate
*.Cnn
Interface Board
Load Application
RS 232
AMPL source code on floppy, disk, etc.
HS link
Target Systems
Advant Controller 100 Series system
Advant Controller 400 Series system

Figure B-1. Data Flow on the Engineering Station

NOTE

In case of node types explicitly supporting application retrieval, such as AC 80 2.0, the Function Chart Builder offers also the functions “upload application” and “backtranslate target code”.

Appendix C Glossary

Address
The components of Advant Engineering uses the following address information:
In case of Advant Controller 110 2.0
Station: 0 - 7; module position: 1 - 20,
101 - 120 and 201 - 220 if a subcarrier module is in the subposition 1 and 2, respectively

In case of Advant Controller 110 2.1 and higher and Advant Controller 160
Bus number: 0 - 255; Station number: 0 - 80;
Module position: 1 - 20 (and for I/O stations:101 - 120, …, 701 - 720);
Subposition (position of the subcarrier module): 0 - 2

In case of Advant Controller 70 1.0
Station: 0; module position: 1 - 12

In case of Advant Controller 70 1.1 and higher
Bus number: 0 - 255; Station number: 0 - 80;
Module position: 0 - 12 (and for I/O stations:100 - 112, …, 700 - 712);
Subposition: not relevant

In case of Advant Controller 55 1.0
Unit: 0 - 4; second part of address: 0 (not relevant)

In case of Advant Controller 55 1.1
Bus number: not applicable; Station number: not applicable;
Unit number: 0 - 4; Subposition: not applicable

Addressing for Advant Controller 80 is typically the same as for Advant Controller 70 1.1.

Advant Controller: A computerized process control system manufactured by ABB (Asea Brown Boveri).

Advant OCS: The term “Advant OCS” is used for Advant Open Control Systems with Master software. Open Control Systems (OCS) is used to describe the new generation of automation systems.

AF 100: Abbreviation for Advant FieldBus 100.

AMPL: ABB MasterPiece Programming Language. Uses text format to specify MasterPiece data base and process control programs.

APB: Abbreviation for Application Builder.

Application Builder: A component of AMPL Control Configuration and AdvaBuild for Windows running on Advant Station 100 Series Engineering Station. It serves as environment for all other components, and provides functions for multi-project administration and project and data maintenance.

Application program: A general concept of an assembly of program functions aiming to realize and to automate an addressed process control functions. Consists of a data base and a PC part. The PC part (process control part) may be structured into different PC programs.

Attribute: A term used for the following purposes:

- To differentiate modifying DB elements of type circuits and circuits (**MOD**) in the DB Section window of type circuits and circuits
- To differentiate DB elements implicitly created by type circuits, remote I/O elements like AI810, and extended DB elements like AOS (**T**, **R**, **X**, respectively) in the **DB Section** window of nodes
- To define the form of the graphical symbol, which represents the instantiated type circuit or the located circuit in the function chart (**Adjust**, **Blank Line**, **Waist**)
- To reflect to what kind of terminals a TC/CI terminal is connected within the type circuit or circuit (**Dynamic**, **DB Reference**, **Non-Invertible**)
- To reference configured AF 100 communication interfaces, and so on, of an AC 100 Series target of newer node type, such as AC 160, AC 110 2.1 and higher, AC 80, AC 70 1.1 and higher, and AC 55 1.1, reported by the **Target | Report Attributes** command.

Backtranslate: Generate the Function Chart Builder's internal data representation in form of a *.ODB file from AMPL source code or its derivatives.

Blocked: The status of an application program in an Advant Controller 100 Series target is blocked if its execution is inhibited by the **Target | Block Program** command.

Call parameter: Parameter that affects the operation of a PC element. For example, the **cyclicity in milliseconds** call parameter sets the cycle time of the program element execution in milliseconds.

Callname: The name that identifies the function of a PC or DB element, of type circuits or circuits. For example, the call name of a control module is CONTRM.

Configuration: This term is used within this manual for the creation and modification of DB elements (configuration from a software point of view).

Configuration part: The DB elements of the user application program.

Convert: A synonym for the following actions:

- Backtranslate
  from AMPL source code to Function Chart Builder internal representation
- Generate
  AMPL source code from Function Chart Builder internal representation
- Generate
  target source code from Function Chart Builder internal representation.

Data base: The MasterPiece data base contains all information concerning connections to the hardware (I/O modules, signals, and so on), connections between PC programs (DAT elements), and connections to operator stations.

Data base element: The view of the MasterPiece data base in AMPL source code is comprised of data base elements. Data base elements (DB elements) are available for I/O modules, I/O channels, data value storage, and so on.

DB: An abbreviation for data base.

DB element: An abbreviation for data base element.

Deblocked: The status of an application program in an Advant Controller 100 Series target is deblocked if its execution is released with the **Target | Deblock Program** command.

Diagram: Graphical and/or text presentation of programs. The Function Chart Builder provides function chart and tree diagram as two different kinds of diagrams for PC programs. They can be displayed and printed.

Disintegrate: To disintegrate instantiated type circuits during backtranslation of source code means that the Function Chart Builder no longer points out the type circuit references in the PC program of the node. Disintegration of a certain class of type circuits is a prerequisite for Advant Controller 100 Series target code generation.

Display: The **Test | Single Display** and **Test | Cyclic Display** commands are part of the Function Chart Builder's on-line test functions available for Advant Controller 100 Series applications. They serve to monitor values of PC terminal variables on-line.

DOS: Abbreviation for Disk Operating System, the program that controls the operation of IBM-compatible personal computers.

Download: To transfer data (for example, target code) from the Function Chart Builder to an on-line connected target.

Executable unit: The PC program can be divided into executable units. An executable unit contains only one structure element (with specified cycle time) or one structure element and one or more subordinate PC elements.

Extended DB element: Complex DB elements that encapsulate channel elements, allow definition of signals independently of the hardware, also for HMI (MMI) purposes, support signal distribution on the AF 100, and if applicable, support also data exchange between redundant CPUs in a station.

FCB: Abbreviation for Function Chart Builder.

Force: The **Test | Activate Force Values** and **Test | Activate All Force Values** commands are part of the Function Chart Builder's on-line test functions, available for AC 100 Series applications. They force values of PC and DB terminals on-line.

Function chart: Diagram for the representation of process control programs. It contains graphical symbols (rectangles) for PC elements and lines for connections between element terminals.

Function Chart Builder: A component of AMPL Control Configuration and AdvaBuild for Windows running on Advant Station 100 Series Engineering Station. It serves for editing, translating, target controlling, testing, and documenting of user application programs.

HMI: Abbreviation for Human-Machine Interface.

Input terminal: Corresponds to one of the input variables of the function represented by the parent element. See also **terminal**.

Instance: An appearance of, or a reference to, a PC element or DB element or type circuit of a certain type. The reference is accompanied by actual parameters. A type circuit can be instantiated, that is, referenced and supplied with parameters, several times.

Instance name: The name of a single PC or type circuit given at its insertion in a PC program. The name of a single data base element given at its creation for the data base of a node.

Internal data representation: All data entered via the Function Chart Builder for a node, node structure, circuit, or type circuit are stored internally in binary representation in files with the extension ODB.

Invert terminal: Inverting the logical sense of the input to a binary input terminal of a PC element.

Item designation: Item designations are used for identification of DB elements, PC elements, located circuits, and instantiated type circuits of an application.

MasterPiece: A computerized process control system manufactured by ABB (Asea Brown Boveri).

MFB: Abbreviation for Master Fieldbus.

MMI: Abbreviation for man-machine interface.
Use the term Human-Machine Interface (HMI) instead.

MP 200/1: Abbreviation for Masterpiece 200/1.

Node: Physically, a node is a process station of a process control network. The final target of application programs prepared by the Function Chart Builder is a (physical) node, that is, an Advant Controller or a MasterPiece.

In the Function Chart Builder, node denotes a program concerning a physical node. It has all references resolved, the referenced parts are expanded, and it is fully checked. A node consists of process control elements (PC elements) and data base elements (DB elements), and of type circuits resolved at instantiation.

Off-line: The counterpart of on-line.

On-line: A Function Chart Builder operation is on-line if it involves direct data transfer to and/or from a connected Advant Controller 100 Series target.

On-line Builder: A component of AMPL Control Configuration and AdvaBuild for Windows running on Advant Station 100 Series Engineering Station. It provides functions for loading and dumping AMPL source code to/from Advant Controller 400 and MasterPiece 200/1 targets and for controlling them.

ONB: Abbreviation for On-line Builder.

Output terminal: Corresponds to one of the output variables of the function represented by the PC element (or type circuit or circuit). See also **terminal**.

PC: Abbreviation for both personal computer and process control.

PC element: Abbreviation for process control element. Part of a process control program. PC elements are represented by rectangles in function chart and by corresponding callnames in tree diagram.

PC program: Abbreviation for process control program.

Programming: This term is used within this manual for the creation and modification of PC programs including connections within the PC program and to DB elements.

Program part: The PC elements of the user application program and their connections (the PC program).

Project: The job of programming a process control system covering several Advant Controller or MasterPiece nodes. Also used to describe all the disk directories and data files used to create the process control data base and programs.

Source code: Program code in text format. The Function Chart Builder uses AMPL source code and its derivatives to specify data base and process control programs. See **AMPL**.

Structure element: PC elements structuring PC programs such as PCPGM, CONTRM, FUNCM, MASTER, SEQ, and SLAVEM. Only they can have subordinate PC elements.

Target: A physical location in an Advant Controller or MasterPiece process control network. A target or (physical) node usually contains an intelligent device like an Advant Controller or a MasterPiece.

Target code: Code generated by the Function Chart Builder that can be loaded directly to an Advant Controller 100 Series target and executed there.

Terminal: Connection point of an element.

A PC terminal is a short line leading to or from the graphical symbol of a PC element (or instantiated type circuit or located circuit) in the function chart. It corresponds to one of the input or output variables of the function represented by this element.

A DB terminal is a short line leading to or from the graphical symbol of a DB element in the Edit DB Terminal Values dialog box. It corresponds to one of the input or output variables (properties) of the function represented by the DB element.

Translate: A synonym for the following actions:

- Backtranslate
  from AMPL source code to Function Chart Builder internal representation
- Generate
  AMPL source code from Function Chart Builder internal representation
- Generate
  target source code from Function Chart Builder internal representation.

Tree diagram: Diagram for the representation of process control programs. The tree diagram shows the process control programs’ structure.

Type circuit: A standard solution for a special process control problem, usually well tested and documented. The Function Chart Builder supports the creation, code generation, and instantiation of type circuits (TCs).

A type circuit consists of symbol declarations, instructions creating or modifying DB elements, and PC elements. A type circuit can be instantiated several times within a node, node structure, or circuit. By assigning different values to its parameters at instantiation, you vary the function of each type circuit instance.

Upload: To transfer data (for example, parameter values) from an on-line connected target to the Function Chart Builder.

User application program: The user-written part of the software for an Advant Controller node consisting of the DB elements (configuration part) and the PC elements (program part).

Symbols
*.C?? 8-3
*.C00 8-3
= (equal) 5-31

A
About On-line and Off-line States 9-1
AC 100 Series 1-6
AC 110 1-6
AC 160 1-6
AC 400 Series 1-6
AC 55 1-6
AC 70 1-6
AC 80 1-6
Acknowledge logic of MMI connection of extended DB elements 4-64
ACT terminal 4-16, 4-20, 4-65
Activity type of step 5-33
Advant Controller 100 Series 1-6
Advant Controller 110 1-8
Advant Controller 160 1-8
Advant Controller 400 Series 1-6
Advant Controller 55 1-9
Advant Controller 70 1-9
Advant Fieldbus 100
    communication via DataSet 4-13, 4-17
    routing 2-4
Advant Station 100 Series Engineering Station 1-6, 1-9
    data flow in B-2
AF 100 1-6
AF100 OPC 1-7
AMPL source code 3-1
APB 1-7
Application
    retrieval 8-3
Application Builder 1-7
Application Checksum 7-14
Application examples A-1
Automatic creation of DSP/DAT/EVS(S)/MDAT 4-64

B
Backtranslate Source 3-1, 6-3
Backtranslate Target Code 3-2
Backup user application program 3-3
Base software 8-4
Basic (PC) element 5-19
BCB 1-7
BLOCK element 5-24
Block Program 8-2
BLOCKED terminal 4-16
Boolean 5-4
Bus Configuration Database 1-7
Bus number 2-2
BUS terminal 4-16

C
CALC_ERR terminal 4-67
CALC_VAL terminal 4-56, 4-66
Call name
    of DB element 4-3
    of PC element 5-14
Channel 4-16, 4-20
Checksum 7-14
Clear PROM 8-2
Client engineering station 2-5
Common
    parameter 5-11
    variable 5-10
Communication
    between redundant CPUs 4-53
    between stations by extended DB elements 4-48
    via Advant Fieldbus 100 4-13, 4-17
    via RCOM 4-19
Computer network 2-5
Conditions of successful connection 2-7
Configuration 1-2
    by DB elements 4-2
    of DB elements describing common data 4-9
    of DB elements describing the hardware 4-6
    of Event Sets 4-33
    off-line 6-1
    on-line 9-5
    part 4-1
Connect AS 100ES to an AC 100 Series Target 7-2
Connect controller to engineering station 7-1
Connection
    between two PC elements 5-27
    of constants to PC terminal 5-29
    of extended DB elements to PC elements 4-63
    of extended DB elements to signal sinks 4-60
    of PC element to DB element 5-28
Constant 5-10
CONTRM control module 5-22

Control
  of car position (example) A-7
  of lift (example) A-1
  of slaves in RCOM net 4-23
Controller 1-7
Conventions 1-2
Convergence (of sequence branches) 5-45
Convert source code 6-3
CRC 7-14

D

DAT element 4-10
Data
  flow in AS 100ES B-2
  transmission via RCOM 4-23
  type 5-2
DataSet communication 4-13
DB 1-7
DB elements 4-2
Deblock Program 8-2
Default user name 4-11
Design flow of AC 100 Series installation 2-1
Diagnosis program 8-6
Diagnostics program 8-6
Direct HW access with extended DB elements 4-71
Disconnect controller from engineering station 7-14
Display
  DB terminals 10-2
  PC terminals 10-4
Divergence (of sequence branches) 5-41
Documentation 1-3
DS_COM terminal 4-52, 4-61, 4-68
DSP_IDNT terminal 4-68
DSP_REF terminal 4-68
DSP_STAT terminal 4-68
Dynamic input 5-18

E

Engineering board 1-7
Engineering station 1-7
  client 2-5
  server 2-5
EV_COM terminal 4-58, 4-69
Event
  detection 4-23
  handling in AF100 or RCOM net 4-23
  handling with extended DB elements 4-58
  transfer 4-25
Event Set 4-33
EVS_IDNT terminal 4-69
EVS_REF terminal 4-69
Example
  control of car position A-7
  control of lift A-1
  extended DB element configuration 4-72
Exchange data 4-17
Executable unit 5-2
Execution
  of elements 11-5
  of on-line changes 11-7
  order (of controller software) 11-4
Extended DB element
  acknowledge logic of MMI connection 4-64
  automatic DB element creation 4-64
  communication between stations 4-48
  connection to PC elements 4-63
  direct HW access 4-71
  event handling 4-58
  example 4-72
  overview 4-44
  processing 4-56
  source selection 4-49
  special terminals 4-65
  status handling 4-58

F

FCB 1-7
FIFO 4-39
Files in AS 100ES B-1
FILT_FTR terminal 4-35
Flash EPROM 8-2
Force
  DB terminals 10-3
  PC terminals 10-5
FUNCM module 5-22
Function
  diagram of sequence 5-32
  parameter 5-10, 5-16
Function Chart Builder 1-7

G

Generate Source 6-3
Generate Target Code 3-2, 6-3
Generate user application program 3-1
Graphic symbol
  of DB element 4-5
  of PC element 5-12
Group data 5-8

H
Handling of type circuits 6-2
Hardware description of AC 100 Series 4-2
HI_LIM terminal 4-67
HOLD function (in sequence) 5-52
HW_PCDIA terminal 4-70
HW_REF terminal 4-50, 4-60, 4-65
HYST terminal 4-34

I
I/O station 4-2
IDENT terminal 4-16
Identification
    of base software 8-4
    of options 8-4
Identification of PC elements 5-25
In-line code 6-4
Input and output phases 11-3
Input of PC element 5-9
Integer 5-5
Internal software of AC 100 Series 11-1
Inverted input 5-19
Item designation
    of DB element 4-5
    of PC element 5-13

L
LIM_SRC terminal 4-71
Load Application 8-1

M
Manual organization 1-1
Master-slave control module 5-23
MDAT element 4-12, 4-53
MDAT_COM terminal 4-53, 4-62, 4-69
MDAT_ID terminal 4-70
MMI_USE terminal 4-70
Modify the application program on-line 9-6
Monitor
    DB terminals 10-2
    PC terminals 10-4

N
Name of DB element 4-6
Net number 2-2
Node 1-2, 2-2
    number 2-2
    setup on AS 100ES 2-2
NORM_TR terminal 4-33, 4-35

O
ODB format 3-1
Off-line
    configuration 6-1
    programming 6-1
On-line
    configuration 9-5
    editing
        rules 9-6
    programming 9-5
Options 8-4
Output of PC element 5-9
Overview of extended DB elements 4-44

P
PARDAT element 4-10
Password 7-11
PC diagram 5-2
PC elements 5-1
PC Instance Names 5-15
PC program 5-1
PC programming 5-25
PCPGM program module 5-21
Point-to-point connection 2-4
Preparation
    of connection 2-7
    of engineering station 2-7
    of intermediate nodes 2-8
    of links 2-7
Process I/O handling in controller 11-3
Processing in extended DB elements 4-56
Program
    module 5-21
    structured 5-20
Programming 1-2
    off-line 6-1
    on-line 9-5
Project setup on AS 100ES 2-2
Protection 7-11

Q
Q_EMPTY terminal 4-39, 4-41

R
RCOM (Remote COMmunication) 4-19
    routing 2-4
Real 5-6

REF1 - REF32 terminals 4-28
Release history 1-5
Remote access possibilities 2-5
Remote access protection 7-14
Report Attributes 8-5
Report Options 8-4
Report Status 8-4
Reset execution 11-6
Restart 8-2
Restore user application program 3-3
Restriction of network programming 7-14
Restrictions at generating source code 6-4
Retrieve Application 3-2
REVACT function parameter 5-16
Rotary switch 2-2
Routing
    via Advant Fieldbus 2-4
    via RCOM 2-4
Rules
    of editing on-line 9-6

S
Save in PROM 8-2
Save user application program 3-3
SCALING terminal 4-72
Scan factor 4-19
Sequence 5-32
    function diagram of 5-32
    selection 5-47
    single 5-39
    with loop 5-51
Sequential control 5-23
Series 1-7
Server engineering station 2-5
Set Date and Time 8-2
Set DB Terminals 10-3
Signal feedback 5-27
Single sequence 5-39
Sort order 10-1
Source code
    files B-1
    restrictions 6-4
Source selection for extended DB elements 4-49
SOURCE terminal 4-65
Special terminals of extended DB elements 4-65
Station number 2-2
STATION terminal 4-16
Status handling with extended DB elements 4-58
STATUS terminal 4-51, 4-60, 4-67
Step activity type 5-33
Store user application program 3-1
Structure
    of internal controller software 11-1
    PC element 5-20
Structured program 5-20
Switch to on-line mode 9-5
Symbolic name for variable or constant 5-11

T
Target 1-2
    code 8-3
    password 7-11
Target Access Protection 7-11
Target Password Protection 7-11
Terminal
    name 4-5, 5-17
    number 5-17
    of PC element 5-17
    type 5-18
Terminology 1-6
Test functions
    configuration part 10-1
    program part 10-3
Thumbwheel switch 2-2
Time
    as data type 5-7
    synchronization in RCOM net 4-23
Transfer of time-tagged events via RCOM 4-23
Translate functions 6-3
Type circuit handling 6-2

U
Update Slave 8-2
Update status report 8-4
Upload 3-2
Use sequences 5-32
User application program 3-1
USER terminal 4-16

V
V.24 interface 1-9
VALUE terminal 4-33, 4-56, 4-66

W
Windows 1-9

3BSE 009 626R401

December 1999